НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Система відображення стану доріг»

Виконав:
студент IV курсу, групи ІО-63
Диренков Владислав Олексійович _____

Керівник:
асистент кафедри ОТ
Стешин Віктор Васильович _____

Консультант з нормоконтролю:
Професор кафедри ОТ, д.т.н.
Сімоненко Валерій Павлович _____

Рецензент:

_____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Диренкову Владиславу Олекчійовичу

1. Тема проєкту «Система відображення стану доріг», керівник проєкту Стешин Віктор Васильович, асистент кафедри ОТ, затверджені наказом по університету від «07» травня 2020 р. № 1081-с

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту див. технічне завдання

4. Зміст пояснювальної записки дослідження предметної області, огляд існуючих рішень, визначення вимог і завдань для програмного продукту, вибір платформи та технології, обґрунтування оптимальності використання обраних інструментів для розробки, реалізація проєкту.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сімоненко В.П.		

7. Дата видачі завдання _____

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломного проєкту.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Відмітки про виконання
1	*Затвердження теми роботи*	01.09.2019	виконано
2	*Вивчення та аналіз завдання*	02.09.2019-02.02.2020	виконано
3	*Розробка архітектури та загальної структури систем*	03.02.2020-03.03.2020	виконано
4	*Розробка структур окремих Підсистем*	04.03.2020-15.03.2020	виконано
5	*Програмна реалізація системи*	16.03.2020-12.04.2020	виконано
6	*Оформлення пояснювальної записки*	13.04.2020-17.05.2020	виконано
7	*Захист програмного продукту*		
8	*Передзахист*	26.05.2020	
9	*Захист*		

Студент Владислав ДИРЕНКОВ

Керівник Віктор СТЕШИН

Анотація

У бакалаврській дипломній роботі реалізовано систему відображення стану доріг для системи автопілоту , призначеної для забезпечення втопілотовоного транспортного засобу, всією необхідною інформацією для безпечного притримання траєкторії руху.

Програма дозволяє прокладати траєкторію руху транспортного засобу підєднаного до системи відображення стану доріг, що надає можливість транспортному засобу можливість уникнення аварійних ситуацій та прокладення більш безпечного та комфортної траєкторії руху . Програмний продукт був реалізований на мові C++ та Java за допомогою бібліотек Qt Automotive Suite та OpenStreetMaps у візуальному середовищі розробки Qt Creator та JOSM.

Аннотация

В бакалаврской дипломной работе реализована система отображения состояния дорог для системы автопилота, предназначенной для обеспечения втопилотовоного транспортного средства, всей необходимой информацией для безопасного удержания траектории движения.

Программа позволяет прокладывать траекторию движения транспортного средства пидеднаного к системе отображения состояния дорог, предоставляет возможность транспортному средству возможность избежать аварийных ситуаций и прокладки более безопасного и комфортного траектории движения. Программный продукт был реализован на языке C ++ и Java с помощью библиотек Qt Automotive Suite и OpenStreetMaps в визуальной среде разработки Qt Creator и JOSM.

Annotation

The bachelor's thesis implements a system for displaying the condition of roads for the autopilot system, designed to provide the autopiloting vehicle with all the necessary information to safely follow the trajectory.

The program allows you to lay the trajectory of the vehicle connected to the road condition display system, which allows the vehicle to avoid accidents and pave a safer and more comfortable trajectory.The software was implemented in C ++ and Java using the Qt Automotive Suite and OpenStreetMaps libraries in Qt Creator and JOSM visual development environments.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ІАЛЦ.467100.002 ТЗ	Система відображення стану доріг. Технічне завдання	4	
3	А4	ІАЛЦ.467100.003 ПЗ	Система відображення стану доріг. Пояснювальна записка	64	
4	А4	ІАЛЦ.467100.004 А1	Система відображення стану доріг. Лістинг програми		

					ІАЛЦ.467800.001 ВП		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Диренков В.О.			Система відображення стану доріг. Відомість дипломного проекту	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Стешин В.В.					
Реценз.						НТУУ “КПІ”, ФІОТ,ІО-63	
Н. Контр.		Сімоненко В.П.					
Затв.		Стіренко С.Г.					

Технічне завдання
до дипломного проекту

на тему: «Система відображення стану доріг»

Київ – 2020

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується на розробку системи відображення стану доріг.

Область застосування: використання системи в навчальному процесі.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить завдання на виконання розробки системи відображення стану доріг, затвердженою кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний Інститут ім. Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка системи відображення стану доріг.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література з комп’ютерних технологій, публікації в періодичних виданнях, публікації в Інтернеті за даним питанням.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- Можливість розгортання додатка на популярних операційних системах;
- Можливість відображення маркерів на мапі;
- Задання маршруту руху транспортного засобу;
- Можливість додавання нового функціоналу;

5.2. Вимоги до програмного забезпечення

- Операційна система Ubuntu 20.04, Ubuntu 18.04, Ubuntu 16.04, Windows 10

5.3. Вимоги до апаратного забезпечення

- Комп’ютер на базі процесору Intel Pentium 3 і вище

- Оперативної пам’яті не менше 512 Мбайт
- Підключення до Інтернету

6. ЕТАПИ РОЗРОБКИ

	Дата
Затвердження теми роботи	01.09.2019
Вивчення та аналіз завдання	02.09.2019-09.02.2020
Розробка архітектури та загальної структури систем	10.02.2020-10.03.2020
Розробка структур окремих підсистем	10.03.2020-15.03.2020
Програмна реалізація системи	16.03.2020-08.04.2020
Оформлення пояснювальної записки	09.04.2020-15.05.2020
Захист програмного продукту	
Передзахист	26.05.2020
Захист	

Пояснювальна записка
до дипломного проекту

на тему: «Система відображення стану доріг»

Київ - 2020 року

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

GPS	Система глобального позиціювання (англ. Global Positioning System)
OCX	Сегмент оперативного управління (англ. Operational Control Segment)
API	Прикладний програмний інтерфейс(англ. Application Programming Interface, API)
OS	Операційна система (англ. operating system)
VM	Віртуальна машина (англ.Virtual Machine)
JVM	Віртуальна машина (англ. JAVA Java virtual Machine)
LTS	Довгострокова підтримка системи (англ. Long-term support)
SNAT	Статичне перетворення мережових адрес (англ.Static Network Address Translation)
GKE	Google Kubernetes двигун (англ. Google Kubernetes Engine)
NAT	Перетворення мережевих адрес (англ. Network Address Translation)
BSM	
WLAN	Локальна мережа (англ. Wireless Local Area Network)
CAM	Повідомлення про кооперативну обізнаність (англ. Cooperative Awareness Message)
DENM	Децентралізоване повідомлення про навколишнє середовище (англ. Decentralized Environmental Notification Message)
VPC	Віртуальна приватна хмара (англ. Virtual Private Cloud)
JOSM	Редактор мап OpenStreetMap на мові Java (англ. Java OpenStreetMap)
OSM	Сервіс мап (англ.OpenStreetMap)

ВСТУП

На сьогодні розробляється досить багато систем автопілоту автомобіля. Кожна з яких відрізняється одна від одної, але мають багато спільного, система автопілоту автомобіля потрібна для зниження кількості ДТП та зменшення людського фактору під час керування автомобіля.

Актуальність теми

Через втрати та травмування населення під час ДТП та фінансові збитки в економіці країн, автопілот дозволить значно знизити ці показники. Також виникає потреба у доставці вантажу у небезбечні зони. Більшість компаній перевізників бажають розробити автопілот для вантажівок, щоб зменшити затрати на водіїв та їх збиткові помилки.

Мета і задачі дослідження

Мета роботи – розробка системи автопілоту автомобіля який буде базуватися на QT Automotive Suite та системи відображення перешкод на мапі.

Щоб досягти поставленої мети було сформовано наступні задачі:

- Проаналізувати вже існуючі автомобільні автопілоти;
- Розробити модель роботи автопілоту та реалізувати його програмну частину;
- Розробити систему відображення перешкод на мапі;
- Провести віртуальне тестування на різних моделях;
- Отримані результати тестування проаналізувати та дослідити;
- В разі отримання задовільнх результатів провести тестування в умовах доріг загального користування з допомогою водія;

Практичне значення

Автопілот дозволить більш надійно та безпечно переміщатися на автомобілі, також зменшить кількість порушень правил дорожнього руху, що сприятиме зменшенню кількості ДТП та травмувань на дорозі.

Розділ 1

ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ СЕРВІСІВ ВІДОБРАЖЕННЯ СТАНУ ДОРІГ ТА ДОПОМІЖНИХ СИСТЕМ

Система відображення доріг – це система, що відображає на мапі вибоїни, затори, ремонтні роботи та аварійно не безпечні ділянки. Система відображення використовують у розробці адаптивного та автомномного автопілоту, завдяки системі відображення стану доріг автопілоти що керують транспортними засобами, робить планування маршруту більш безпечним, комфортним та коротким для пасажирів.

Програмне робить відмітки стану доріг зчитуючи данні автомобіля, а саме:

- Повертяння керма;
- Зміна передачі;
- Прискоренням та спповільненням;

Сенсори збирають інформацію про дії автопілоту автомобіля, які лягають в основу дій системи відображення стану доріг.

Зазвичай встановлювані датчики для коректної роботи системи :

- Камери;
- Системи глобального позицюювання (GPS);
- Датчики одометра;
- Гіростабілізатори;
- Радари;
- Лідари;

1.1.GPS

Концепція GPS заснована на часі та відомому положенні GPS-спеціалізованих супутників. Супутники несуть дуже стійкі атомні годинники, які синхронізуються один з одним і з наземними. Будь-який дрейф від часу, що зберігається на землі, виправляється щодня. Таким же чином відомі місця супутників з великою точністю. GPS-приймачі також мають годинники, але вони менш стабільні та менш точні.[1]

Для точної навігації для GPS потрібні чотири або більше супутників. Рішення навігаційних рівнянь дає положення приймача разом з різницею між часом, який зберігається за бортовим годинником приймача, і справжнім часом доби, тим самим виключаючи необхідність більш точного та, можливо, непрактичного годинника на основі приймача.[1]

1.1.1Структура GPS

GPS складається з трьох основних сегментів. Це космічний сегмент, сегмент управління та сегмент користувача. Супутники GPS передають сигнали з космосу, і кожен GPS-приймач використовує ці сигнали для обчислення свого тривимірного розташування (широти, довготи та висоти) та поточного часу.

- Космічний сегмент складається з 24 до 32 супутників або космічних транспортних засобів на середній орбіті Землі, а також включає адаптери корисного навантаження до підсилювачів, необхідних для запуску їх на орбіту. Орбіти розташовані так, що щонайменше шість супутників завжди знаходяться в межах зору зору, звідусіль на поверхні Землі. На орбіті кожен космічний транспортний засіб робить дві повні орбіти кожного бічногореалію , повторюючи одну і ту ж наземну доріжку щодня. Станом на лютий 2019 року в спутниковому

групуванню GPS є 31 супутник , 27 з яких використовуються в певний час, а решта виділяються у режимі очікування.[1]

Більше спущених супутників перебувають на орбіті і доступні як запасні частини. Додаткові супутники, яких понад 24 покращують точність обчислень GPS-приймача, забезпечуючи надмірні вимірювання. Зі збільшенням кількості супутників у супутниковому угрупуванню було змінено на неоднорідне розташування. Таке розташування показало, що підвищує точність, але також підвищує надійність та доступність системи відносно рівномірної системи, коли декілька супутників виходять з ладу. Наочний приклад 24 супутникового угрупування GPS у русі з обертом Землі зображено на Рис 1.1, Рис 1.2, Рис 1.3 де чорні крапки - супутники які не потрапляють у поле зору, червоні крапки - супутники які потрапляють у поле зору.[1]

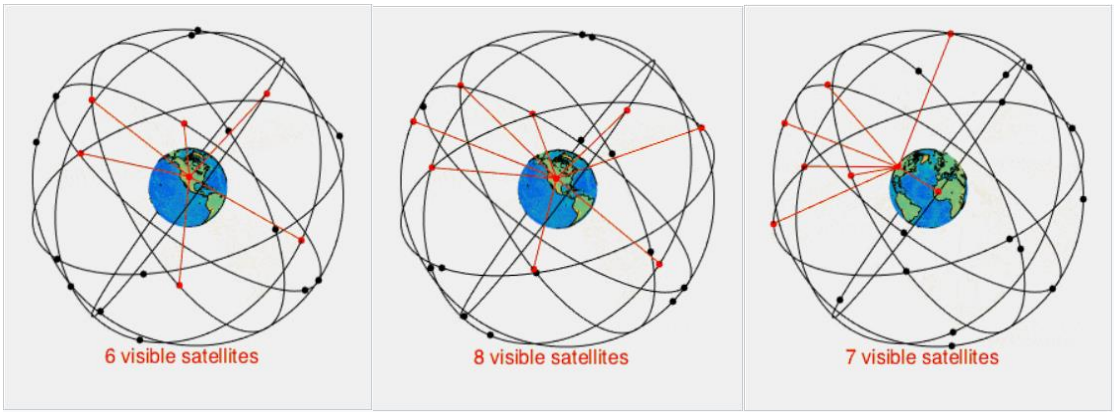


Рис. 1.1. Рис. 1.2. Рис. 1.3.
Наочний приклад супутникового угрупування [10].

- Сегмент управління складається з: головної станції управління, альтернативної головної станції управління, чотирьох виділених наземних антен та шести виділених моніторних станцій. Головна станція також може отримувати доступ до наземних антен Супутникового управління та станцій моніторингу. Шляхи польотів супутників відстежуються спеціалізованими станціями моніторингу космічних сил. Супутникові маневри не є точними за стандартами GPS

- тому для зміни орбіти супутника супутник повинен бути позначений нездоровим , тому приймачі не використовують його.

Після супутникового маневру інженери відстежують нову орбіту від землі, завантажують нові ефемериди і знову відзначають супутник здоровим.[1]

- Користувацький сегмент складається з сотень тисяч військових та сотень мільйонів цивільних, комерційних та наукових користувачів. Взагалі GPS-приймачі налаштовані на частоти, що передаються супутниками, приймачем-процесорами, та високостабільним тактовим годинником найчастіше кристалічним генератором. Вони також можуть містити дисплей для надання інформації про місцезнаходження та швидкість користувачеві. Приймач часто описується його кількістю каналів: це означає, скільки супутників він може контролювати одночасно. Спочатку обмежившись чотирма або п’ятьма, це поступово зростало з роками.[1]

1.1.2 Система оперативного управління нового покоління

Система операційного управління нового покоління (OCX) - це майбутня версія сегменту управління GPS.

OCX буде командувати усіма модернізованими та застарілими GPS-супутниками, керувати всіма цивільними та військовими навігаційними сигналами та забезпечуватиме покращену кібербезпеку та стійкість для наступних поколінь GPS-операцій. Розробка OCX дотримується поступового підходу.

- Блок 0 - це система запуску та управління, призначена для управління операціями запуску та ранньої орбіти та контрольною орбітою всіх супутників GPS III. OCX Block 0 - це підмножина блоку OCX 1, що забезпечує апаратне, програмне забезпечення та базу кібербезпеки для блоку 1.
- У блоці 1 поля оперативної здатності контролювати всі застарілі супутники та цивільні сигнали, військові сигнали, а також супутники

GPS III та модернізований цивільний сигнал та сигнал авіаційної безпеки польоту.
Крім того, блок 1 буде використовувати базові оперативні можливості управління модернізованими військовими сигналами а глобально сумісним сигнало. Він також повністю відповідає вимогам щодо забезпечення інформаційної / кіберзахисту.

- Блок 2 включає в себе розширені оперативні можливості контролювати вдосконалені можливості модернізованих військових сигналів. Блок 2 буде доставлений одночасно з Блоком.

1.2. Опис предметної області

Сервіс відображення стану догір – це сервіс що моніторить стан доріг на присутність заторів, ремонтних доріг, ДТП, лежачих поліцейських, неправильно припаркованих автомобілей та інш.

Система буде складатися з чотирьох серверів: картографічний сервер, клієнт сервер, сервер взаємодій та backend сервер. Щоб організувати максимально ефективну роботу системи буде використано архітектуру запропоновану середовищем розробки Qt Automotive Suite. Збирання даних буде відбуватися за допомогою автопілоту автомобіля.

1.2.1. V2X Автомобіль до всього

V2X або автомобіль до всього – це передача інформації з транспортного засобу до будь-якої організації, яка може вплинути на нього та навпаки. Ця система включає в себе більш специфічні типи зв’язку, як V2I (транспортий засіб - інфраструктура), V2C (транспортий засіб до хмари), V2V (від транспортного засобу до транспортного засобу), V2P (транспортий засіб - пішохід) , V2H (транспортий засіб - дім), V2G (транспортий засіб до мережі) та V2B (транспортний засіб до споруд) зв’язок зображено на мал.1.4.

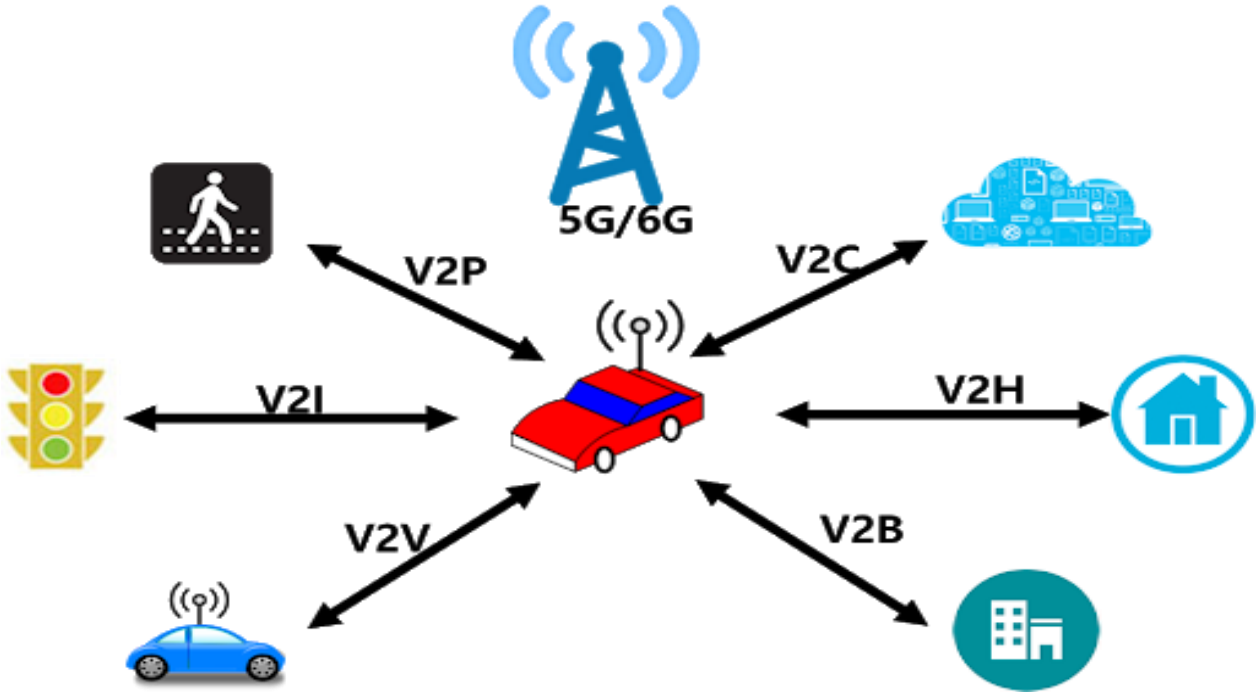


Рис. 1.4 Позначення зв’язку системи V2X.[5]

1.2.3 802.11p (DSRC)

V2X для зв’язку використовує технологію WLAN і пряцює безпосередньо між транспортними засобами та інфраструктурою дорожнього руху, які утворюють автомобільну спеціальну мережу. Якщо два прердавача V2X потрапляють в діапазон один одного, для зв’язку їм не потрібна жодна інфраструктура, що є ключовим для забезпечення безпеки у віддалених або малорозвинених районах. WLAN добре підходить для звязку через низьку затримку, через нього проходить передача повідомлень, як повідомлення про кооперативну обізнаність або основні повідомлення про безпеку та децентралізовані повідомлення про навколишнє середовище.Обсяг даних повідомлень є дуже низьким.

1.2.4 3GPP

3GPP розширив функціональність V2X до підтримки 5G. Ця система включає підтримку прямого зв'язку між транспортними засобами так і традиційну тільникову. Крім того система забезпечує шлях переходу до систем та послуг на основі 5G, що передбачає несумісність та більші витрати порівняно з рішенням на 4G. В той час коли система визначає функції транспортування даних,система не включає семантичний вміст V2X, натомість може запропонувати використати стандарти ITS-G5:

- CAM;
- DENM;
- BSM;

1.2.5 V2C

Підключений автомобіль являє собою автомобіль, який може взаємодіяти з іншими двонаправленимисистемами за межами автомобіля. Це дозволяє автомобілю спільно використовувати доступ до інтернету, дані з інших пристроїв. Для критичних ситуацій для безпеки, автомобілі будуть підключатися за допомогою виділених радіостанцій ближнього зв'язку DSRC. Здатність технології обмінюватися інформацією для додатків транспортного засобу з хмарної системи, дозволяє використовувати інформацію з інших галузей, таких як енергетика, транспорт, розумний дім, які використовують Iot.

Для підключення до Інтернету найчастіше використовується GSM-модуль та телематичні пристрої які інтегровані в автомобільну IT-систему.

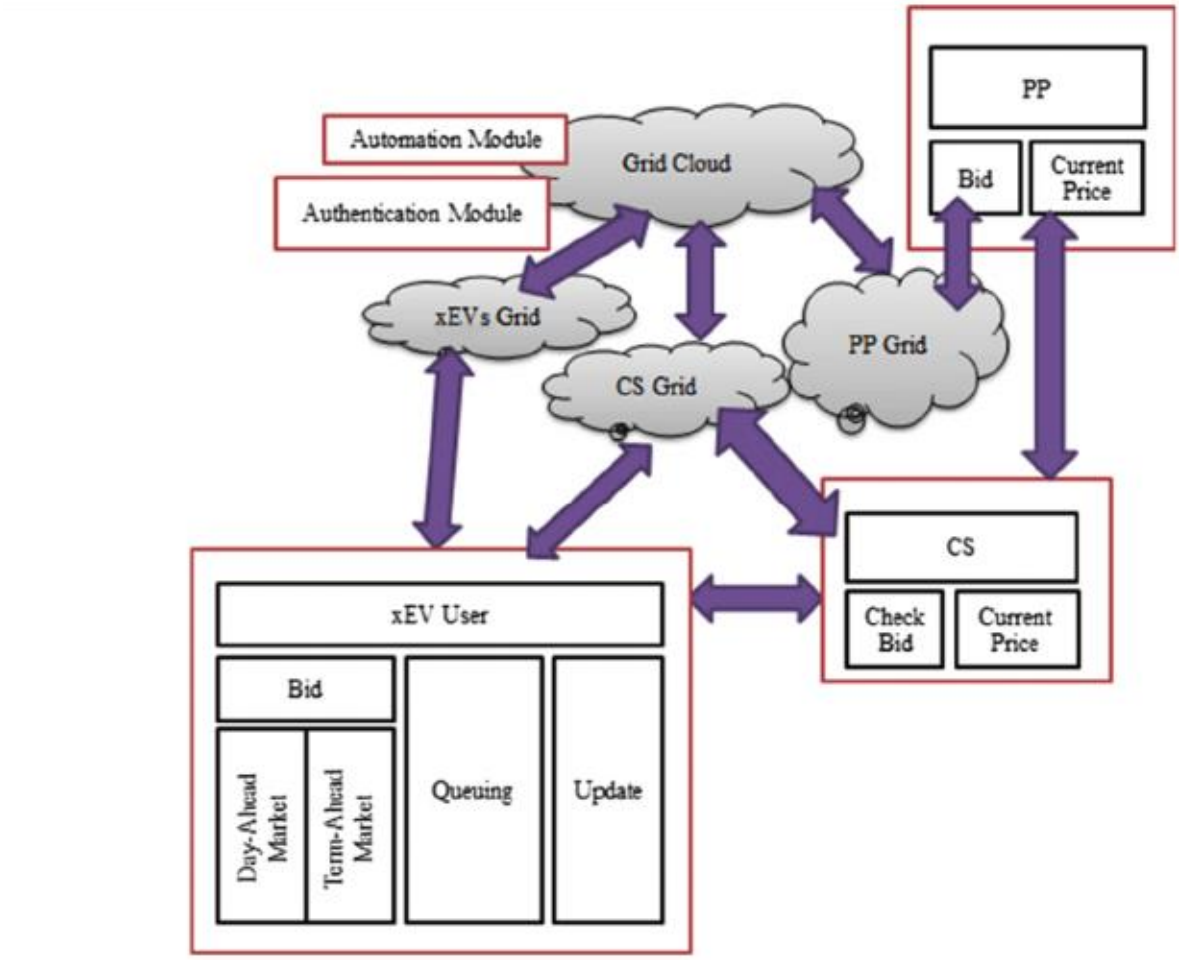


Рис. 1.5 Архітектура V2C.[3]

1.3 ТРАНСЛЯЦІЯ МЕРЕЖЕВИХ АДРЕСІВ NAT

NAT використовується для перетворення IP-адреси комп'ютера або групи комп'ютерів в одину спільну адресу в заголовках пакетів даних, коли пакети відправляються в Інтернет. Оскільки для зовнішнього світу видимий лише одни IP-адрес, NAT забезпечує додаткову безпеку та може мати лише одину публічну адресу для всієї мережі натомість декількох IP-адрес.

Для статично NAT та NAT призначення існує загальний набір правил, які вмзначають набір умов для трафіку та визнчає одну з дій :

- Почтковий інтерфейс;
- Початковиа зона;
- Початковий екземпляр маршрутизації;

Для початкових ніборів правил конфігуруються умови призначення та джерела:

- Початковий інтерфейс, зона або екземпляр маршрутизації;
- Інтерфейс призначення, зона або екземпляр маршрутизації;

1.3.1 Cloud NAT

Система дозволяэ екземплярам віртуальної машини Google Cloud (VM) без зовнішніх IP-адрес та приватних кластерів Google Kubernetes Engine (GKE) надсилати вихідні пакети в Інтернет та отримувати відповідні встановлені пакети вхідних відповідей.[12]

1.3.2Архітектура Cloud NAT

Cloud NAT э досить розповсюджена система, яка визначена програмним забезпеченням. Він не базується на проксі VM або пристроях. Cloud NAT налаштовує програмне забезпечення Andromeda, яке забезпечує вашу мережу VPC так, що воно також забезпечує трансляцію вихідних мережевих адрес (SNAT) для віртуальних машин без зовнішніх IP-адрес, також забезпечує трансляцію адресних мережевих адрес (DNAT) для встановлених пакетів вхідних відповідей.

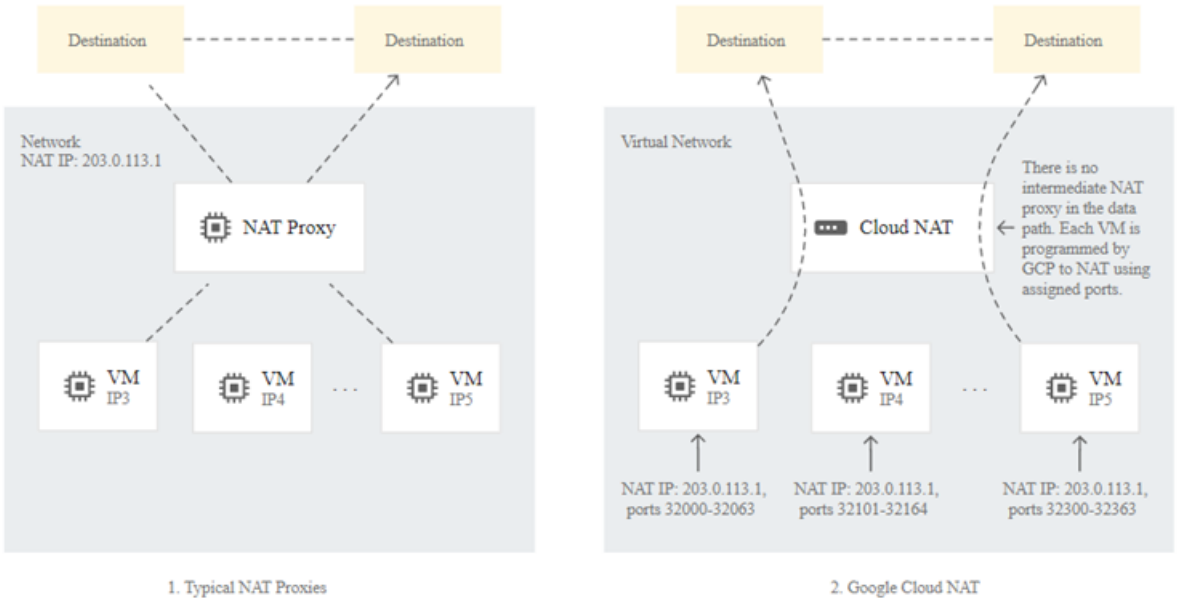


Рис 1.6 Традиційний NAT проти Cloud NAT[13]

1.3.3 Cloud NAT переваги

Cloud NAT має переваги у сферах:

- Безпека:
  Надає можливість зменшити потребу в окремих віртуальних машинахдля кожного зовнішнього IP-адреси.При на лаштуванні шлюзу з’явиться можливість поділитися набором загальних зовнішніх IP-адрес ші стороною призначення.

- Наявність:
  Послуга, що не залежить від віртуальних машин у проекті або від одного фізичного пристрою шлюзу. Налаштувавши NAT-шлюз на хмарному маршрутизаторі, забезпечить площину управління NAT, утримуючи вказані параметри конфігурації.

- Маштабованість:
  Надається можливість налаштування так, щоб автоматично масштабувати кількість IP - адрес NAT які він використовує, та підтримує віртуальні машини , які належать до керованих груп.

- Продуктивність:
  Не зменшується пропускна здатність мережі віртуальних машин. Завдяки мережевої мережі Andromeda.

1.3.4 Система взаємодії GKE

Cloud NAT шлюз може виконувати NAT для вузлів та Pods у приватному кластері. Шлюз повинен бути налаштований так, щоб застосовуватися принаймні до таких діапазонів IP-адрес підмережі для підмережі, яку використовує наш кластер:

- первинний діапазон IP-адрес підмережі
- підмережа вторинного діапазону IP-адрес, що використовується для Pods в кластері
- підмережа вторинного діапазону IP-адрес, що використовується для Служб кластеру

Коли Cloud NAT шлюз налаштований для забезпечення приватного кластера, він резервує адреси джерел та порти джерела для кожного вузла віртуальної машини. Ці адреси джерел NAT і порти джерела можуть використовувати Pods, оскільки IP-адреси Pod реалізуються як псевдонімні IP-діапазони, призначені кожному VM вузла. Процедура резервування порту Cloud NAT резервує щонайменше 1024 адреси джерела та вихідні порти на вузол. Незалежно від Cloud NAT, GKE виинує SNAT використовуючи програмне забезпечення, що працює на кожному вузлі. Cloud NAT може бути корисним і для приватних кластерів, включаючи як локальні VPC, так і кластери на основі маршрутів. Оскільки вузли в неприватному кластері мають зовнішні IP-адреси, пакети, що надсилаються з первинної внутрішньої IP-адреси вузла, ніколи не обробляються Cloud NAT. Однак пакети, надіслані з Pods в неприватному кластері, можуть бути оброблені Cloud Cloud шлюзом, якщо:

- Для рідних кластерів VPC Cloud шлюз NAT налаштований для застосування до вторинного діапазону IP-адрес для Pods кластера;
- Конфігурація IP-маскараду кластера не налаштована для виконання SNAT в кластері для пакетів, що надсилаються з Pods в Інтернет;[13]

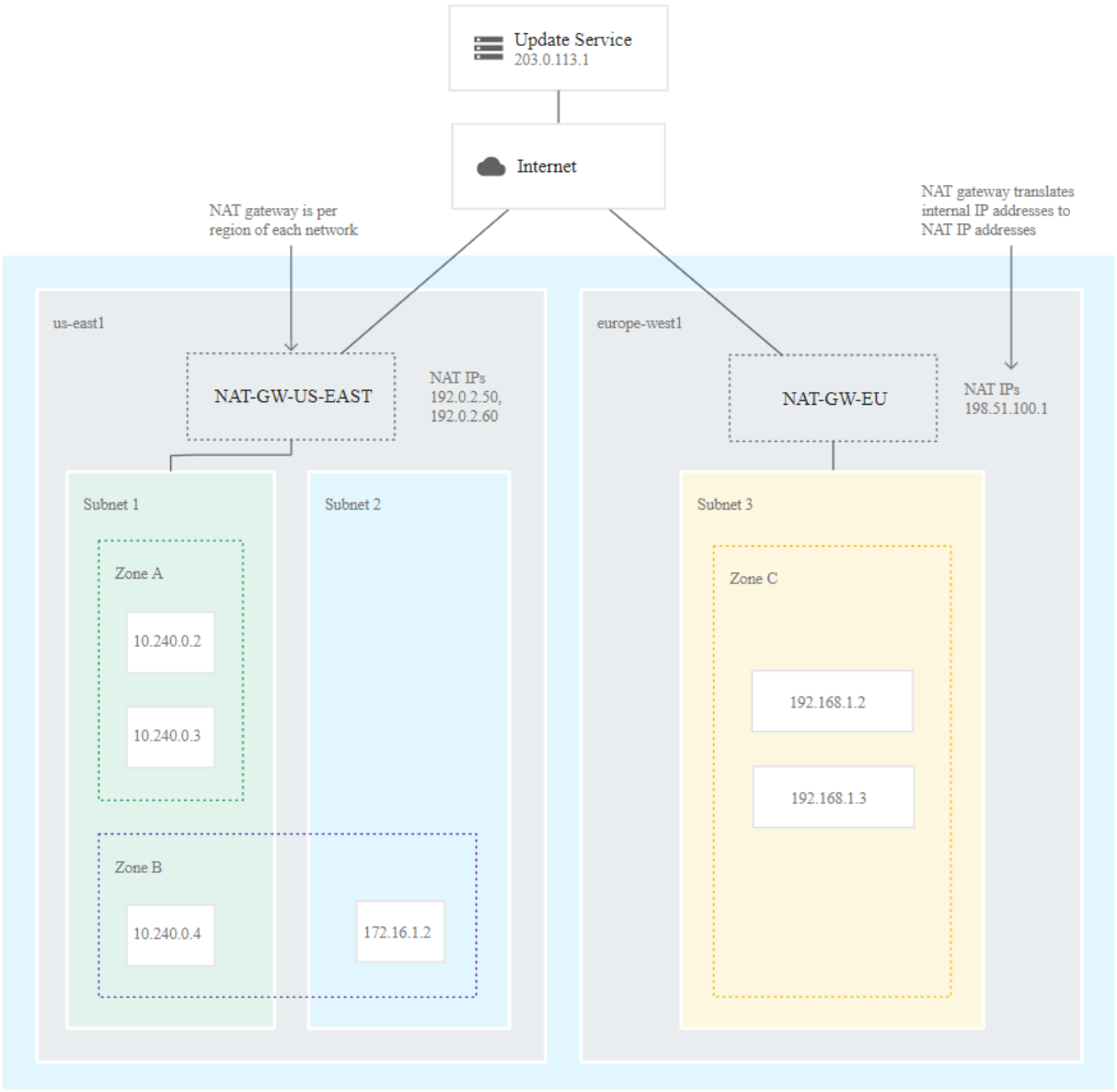


Рис 1.7. Приклад хмари NAT.[13]

В цьому прикладі приклад :

- `nat-gw-us-east` Шлюз виконаний з можливістю застосовувати до первинного діапазону адрес IP з `subnet-1`в `us-`east1регіоні. ВМ якого мережевий інтерфейс не має зовнішній IP – адреса можна відправити трафік в Інтернет , використовуючи або його первинний внутрішній IP – адреса або діапазон IP – псевдонім від первинного IP – адреса діапазону `subnet-1`, `10.240.0.0/16`.

- Віртуальна машина, мережевий інтерфейс якого не має зовнішньої IP-адреси та чия основна внутрішня IP-адреса розташована в `subnet-2` не може отримати доступ до Інтернету, оскільки жоден Cloud-шлюз NAT не застосовується до будь-якого діапазону IP-адрес цієї підмережі.
- `nat-gw-eu` Шлюз виконаний з можливістю застосовувати до первинного діапазону адрес IP з `subnet-3`в `europe-west1` регіоні. ВМ якого мережевий інтерфейс не має зовнішній IP – адреса можна відправити трафік в Інтернет , використовуючи або його первинний внутрішній IP – адреса або діапазон IP – псевдонім від первинного IP – адреса діапазону `subnet-3`, `192.168.1.0/24`.

На рисунку 2.5 ми хочемо, щоб наші контейнери перекладалися на NAT. Щоб увімкнути NAT для всіх контейнерів і вузла GKE, ии повинні вибрати всі IP-діапазони підмережі 1 в якості кандидатів NAT. Не можна NAT лише контейнер 1 або контейнер 2.

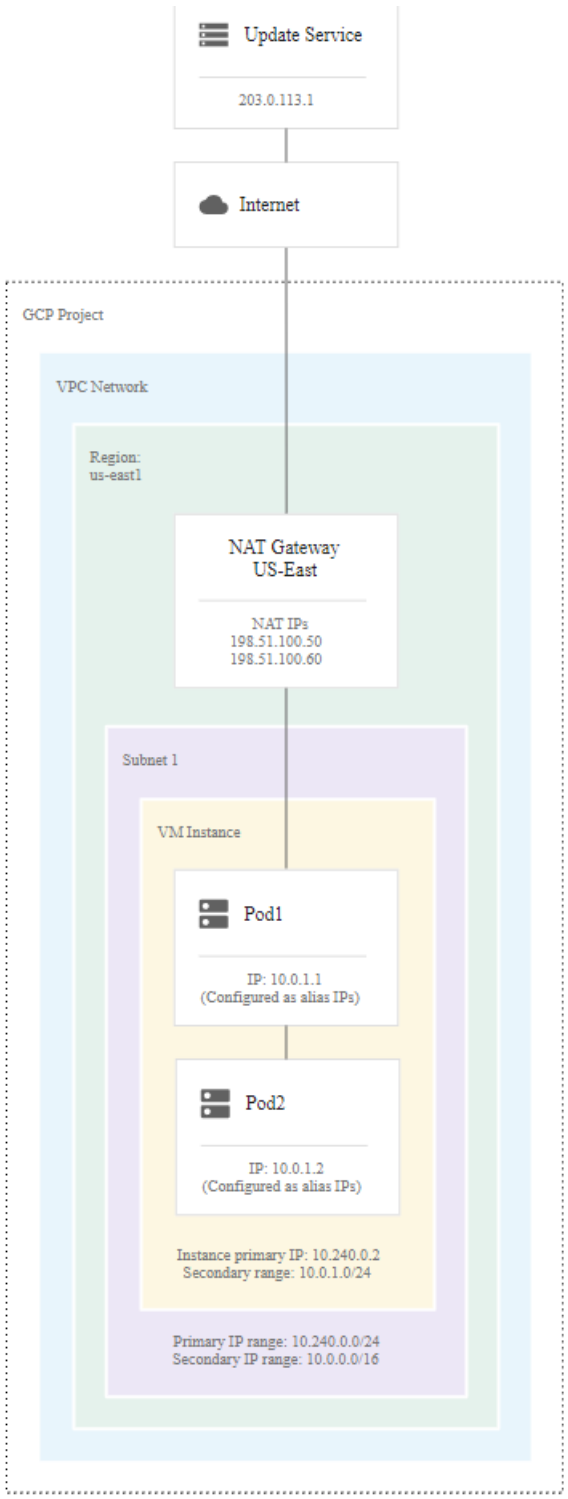


Рис 1.8 Приклад Cloud NAT.[13]

1.5 Технологія OpenStreetMap

Open Street Map використовує топологічну структуру даних з чотирсь основними еементами: [16]

- **Вузли** - це точки з географічним положенням, що зберігаються як координати, пари широти та довготи . Поза способом їх використання вони представляють особливості карти без розміру, наприклад, визначні місця або гірські вершини.

- **Способи** впорядковані списки вузлів , що представляють собою полілінію або багатокутник, якщо вони утворюють замкнутий цикл. Вони використовуються як для представлення лінійних особливостей, таких як вулиці та річки, так і райони, як ліси, парки, парковки та озера.
- **Відносини** - це упорядковані списки вузлів, способів та відносин де кожен член може необов'язково мати "роль" . Відносини використовуються для представлення взаємозв'язку існуючих вузлів і способів. Приклади включають обмеження повороту на дороги, маршрути, які охоплюють декілька існуючих шляхів, наприклад, автомагістраль на великі відстані та ділянки з отворами.
- **Теги** - пари ключових значень. Вони використовуються для зберігання метаданих про об’єкти карти, такі як їх тип, ім'я та фізичні властивості. Теги не є самостійними, але завжди прикріплені до об'єкта: до вузла, способу чи відношення. Нові схеми мітки завжди можуть бути запропоновані шляхом всенародного голосування за письмову пропозицію у вікі OpenStreetMap.

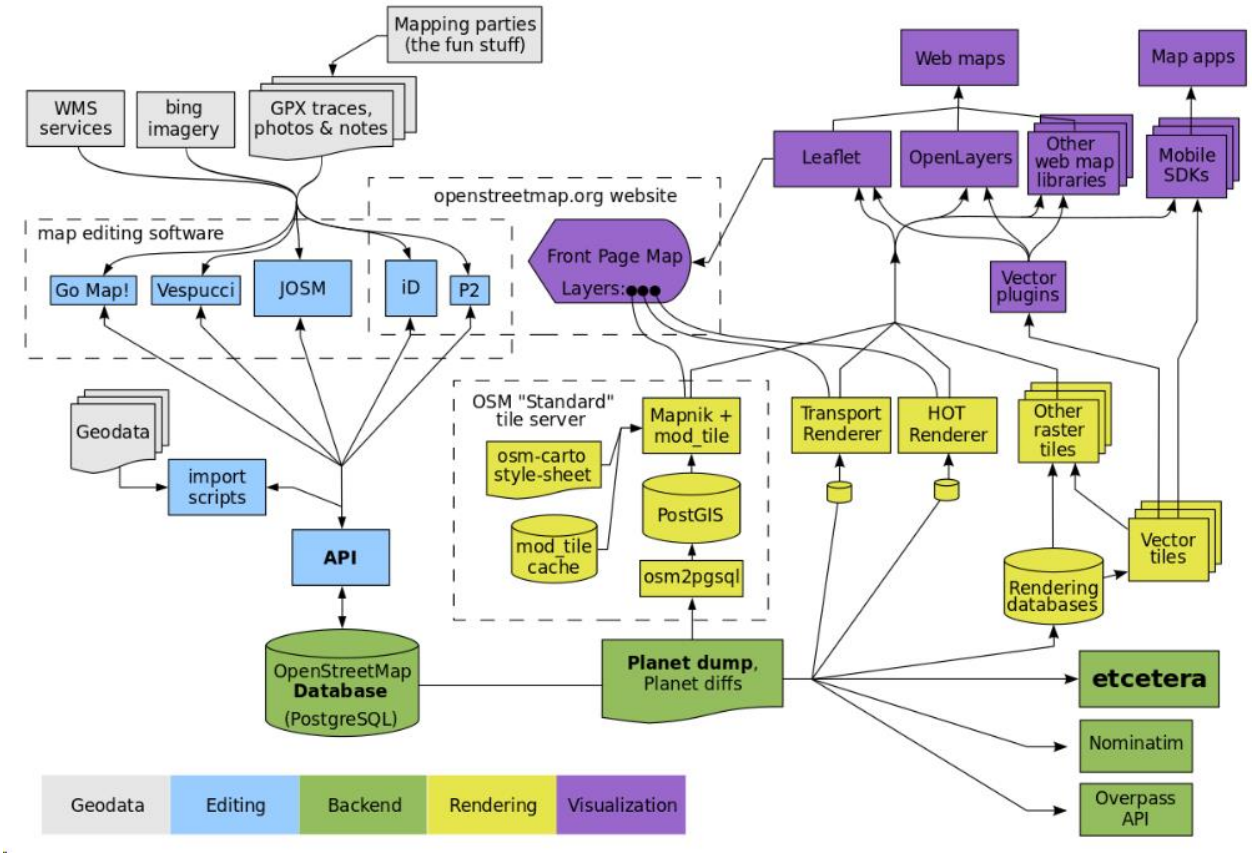


Рис. 1.9 Архітектура OSM[16]

1.5.1. Редактор Java OpenStreetMap

JOSM є інструментом для редагування геоданних OpenStreetMap, редактор має багато додаткофих функцій, але більш складний інтерфейс наж у редактора ID. Цей редактор э вільним програмним інструментом для ПК.

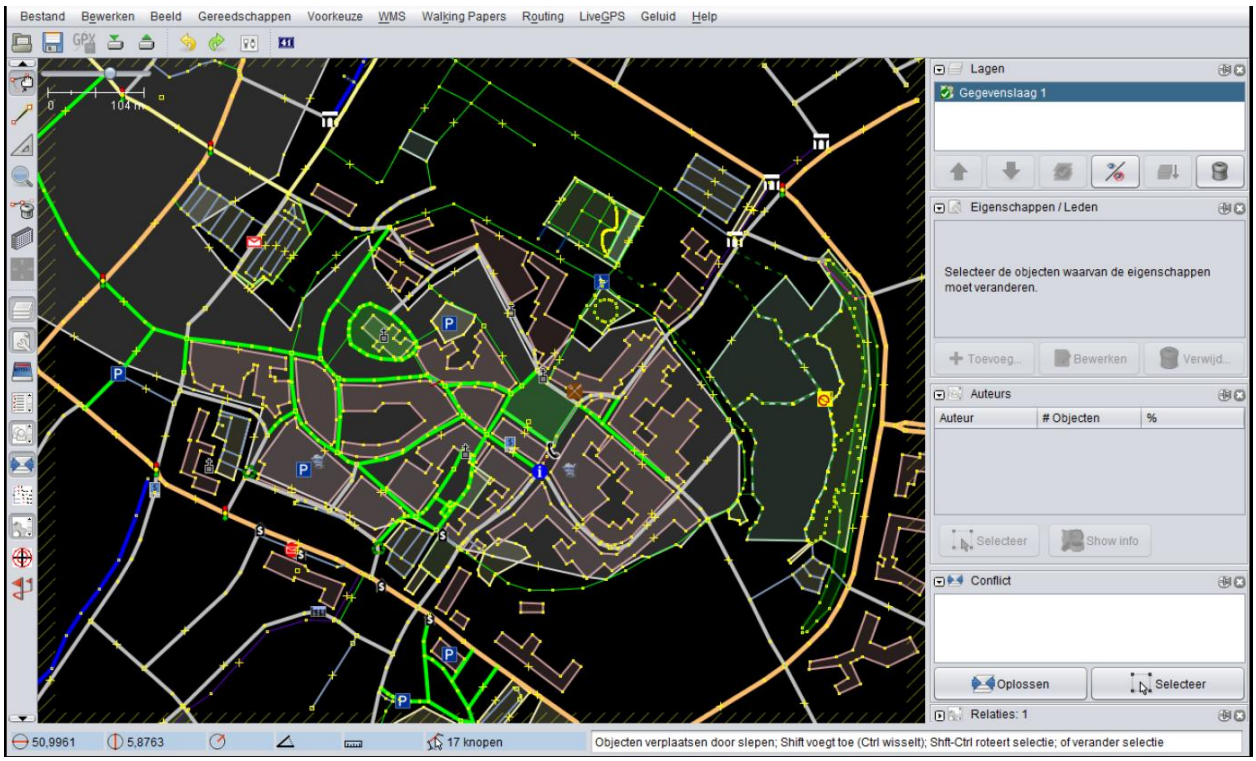


Рис. 1.10 Редактор JOSM[17]

1.5.2. Редактор OSP ID

Онлайн-редактор, створений на JavaScript. Це найпопулярніший редактор OSM за кількістю користувачів.

Особливості iD включають вибір користувальницьких аерофотознімків та нативну підтримку фотографій Mapillary .

Деякі спеціалізовані вилки iD:

- Слайд Strava , який дозволяє легко оптимізувати способи відповідності треків GPS, зібраних користувачами Strava
- iD-Indoor , призначений для картографування в приміщенні
- Mapeo , експериментальний редактор для офлайн-карти у віддалених середовищах
- RapiD , розроблений Facebook як інструмент імпорту для огляду та додавання доріг, виявлених власними алгоритмами Facebook

1.6 Qt Automotive Suite

Середовище розробки для розробки автомобільного програмного забезпечення. Qt Automotive Suite складається з декількох компонентів, побудованих на Qt і Qt Creator .

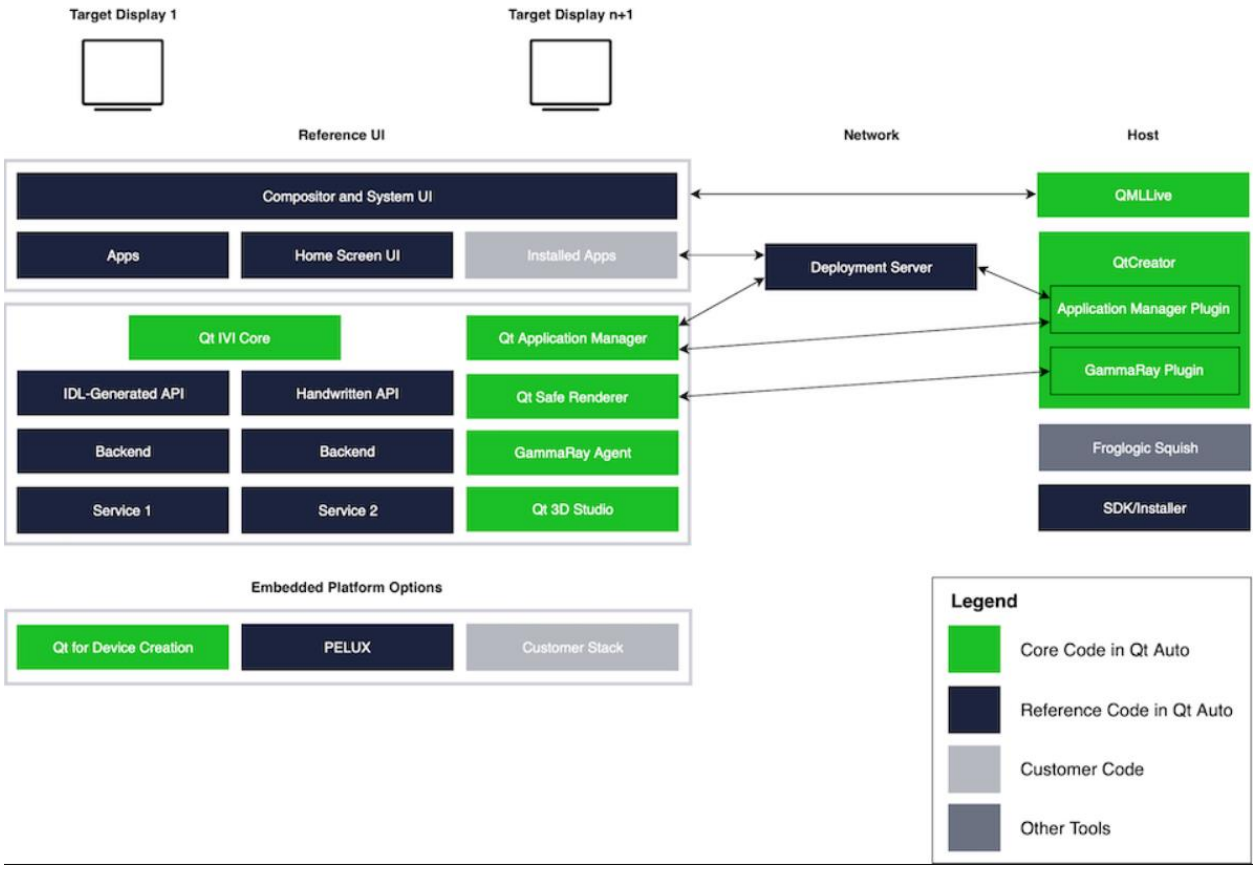


Рис.1.11 Структура Qt Automotive Suite[19]

Компоненти середовища розробки:

- Qt Manager

  Забезпечує основу для системи. Модуль забезпечує керування додатками.

- Qt IVI

  Забезпечує інтерфейси C ++ та QML для доступу до функцій автомобіля та проміжного програмного забезпечення для розваг.

- Reference UI

  Реалізує базовий інтерфейс Qt в системах інформаційних розваг транспортного засобу.
- Qt Safe Renderer

  Візуалізує інтерфейс користувача, для візуалізації індикаторів безпеки, таких як попереджувальні індикатори.

Інструменти:

- QML Live

  Забезпечує живе середовище перезавантажувача для швидкого розвитку додатків Qt Quick, різко скорочуючи час, необхідний для розгортання та тестування змін у процесі проектування інтерфейсу користувача.
- Qt Design Studio

  Середовище розробки та розробки інтерфейсу для швидкого прототипування анімованих інтерфейсів. Це середовище надає інструменти для виконання завдань від процесу розробки додатків, до складання прототипів і, нарешті, до виробництва.
- Qt 3D Studio

  Інструмент, орієнтований на художника, для розробки та побудови багатих користувальницьких інтерфейсів, а також для надання часу виконання на Qt 3D. Цей інструмент дозволяє швидко створювати та прототипувати 2D та 3D інтерфейси користувача. Ви можете скористатися вбудованою бібліотекою матеріалів та ефектів, а також імпортувати об’єкти дизайну з популярних інструментів 3D-дизайну за допомогою обмінних форматів FBX та COLLADA.
- GammaRay

  Плагін для Qt Creator, який забезпечує інтроспекцію програмного забезпечення для програм Qt. Цей плагін допомагає візуалізувати

поведінку програми та маніпулювати нею під час виконання, як локально, так і віддалено на вбудованій цілі.

1.7. Візуалізація мапи

На мапі буде відображатися інформація про місце знагодження автомобіля, будівель, доріг та стану на дорозію. На рисунку 1.12 зображено готовий результат візуалізації.

Рис.1.12 Візуалізація мапи[21]

ВИСНОВОК ДО РОЗДІЛУ 1

У первому розділі було проведено опис предметної області проекту, та аналіз допоміжних систем для вимог до функціоналу додатку та були складені складені основні функції. Також було проведено розбір технологій звязку автопілоту автомобіля V2X, V2G, V2C також було проведено аналіз системи глобального позиціонування. Після проведеного аналізу було зроблено вибір бібліотеки Cmake, Boost, Qt Automotive Suite та системи OpenStreetMap, щоб реалізувати програмний та графічний інтерфейс системи.

РОЗДІЛ 2.

ПРОЕКТУВАННЯ СИСТЕМИ

2.1. Вибір технологій та їх обґрунтування

2.1.1. Вибір платформи для додатку

У розділах 1 та 2 були визначені вимоги щодо проектування системи автопілот для Smart city з системою відображення стану доріг та були поставлені такі завдання:

1. Можливість розгортання додатка на популярних операційних системах;
2. Можливість налаштування швидкості руху;
3. Задання маршруту руху транспортного засобу;
4. Можливість додавання нового функціоналу;

Щоб реалізувати усі вимоги та були дотримані, виникла потреба вибрати зручну платформу для розробки. Вона має бути зручна та досить швидка, щоб зробити систему якомога якіснішою. Серед найбіль популярних систем було виявлено такі комп'ютерні платформи:

- Windows,
- Linux,
- macOS;

Для розробки додатку була обрана Ubuntu 20.04 LTS, яка є Unix подібною тобто відноситься до сімейства Linux. Платформа на даний час є найновішою також даний реліз є LTS тобто вона буде підтримуватися довгий час що дасть змогу вести довгострокову розробку додатку не побоюючись зміни стандартних пакетів які впливають на розробку системи. Також платформа підходить для роботи з обраними у другому розділі технологіями та бібліотеками.

2.1.2. Вибір мови програмування

Для розробки системи автопілот для Smart city була обрана мова C++ Переваги C ++:

- Переносимість - C ++ має портативность або незалежность платформи, що дає можливість користувачеві легко запускати програму на різних операційних системах або інтерфейсах. Ця функція є дуже зручною для Розробників програмного забезпечення.
- Мульти Парадигма - C ++ - Включає логіку, структуру та процедуру програми. Родова, імперативна та об'єктно-орієнтована - це три парадигми C ++.
- Маніпуляція низького рівня - C ++ тісно пов'язаний з C, яка є процедурною мовою, і пов'язаною з машинною мовою, C ++ дозволяє проводити маніпулювання даними на певному рівні.
- Управління пам'яттю - C ++ надає розробнику повний контроль над керванням пам'ятті. Можемо розглядати як як актив, так і як зобов'язання, оскільки йде збільенняє відповідальності користувача за управління пам'яттю. Концепція реалізована за допомогою DMA (динамічного розподілу пам'яті) за допомогою покажчиків.
- Сумісність із C - C ++ в значній мірі сумісний з C. Практично кожна програма без помилок C є дійсною програмою C ++. Залежно від використовуваного компілятора, кожна програма C ++ може працювати на файлі з розширенням .cpp.
- Масштабованість - Масштабованість дає можливість програмі здатність до масштабування. Завдяки цьому програма C ++ здатна працювати в будь-якому масштабі даних. Також з'являється можливість створювати додатки, які потребують досить великої кількості ресурсів.

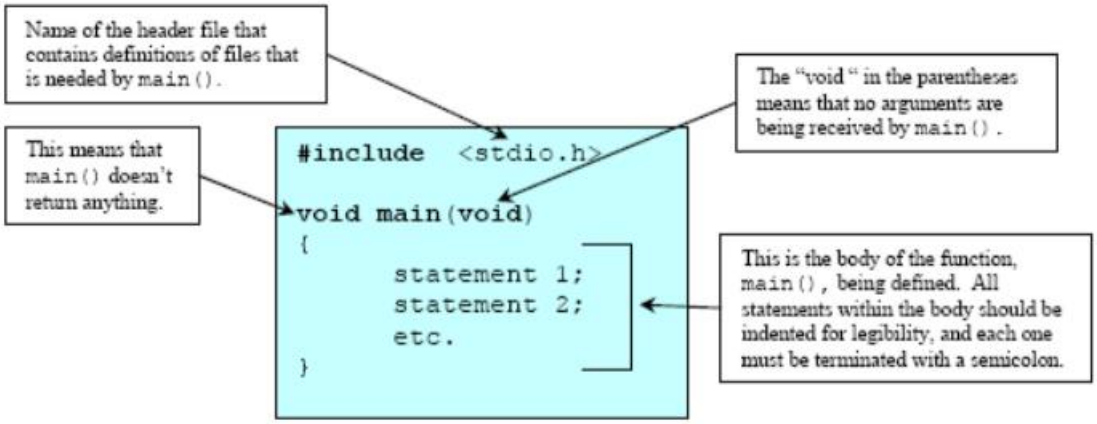


рис. 2.1. Приклад структури C/C++ програми

Для реалізайії OpenStreetMap була обрана мова Java:

Гнучкість . Java довела, що C — поцедурний та залежний від керування та платформи код.

Аплетти. Java додали аплетти — не великі веб-программи, які надають можливість використовувати інтерактивні елементи для візуалізації.

Java TDD або Розробка через тестування. Стандартний спосіб розробки програмного забезпечення.

Об'єктно-орієнтоване програмування.

Концепція, в якій ви не тільки визначаєте тип даних і його структуру, а й набір функцій, що застосовуються до нього. Таким чином, структура даних стає об'єктом, яким можна управляти для створення відносин між різними об'єктами. При ООП можна групувати ці змінні і функції за допомогою контексту, маркувати їх і посилатися на функції в контексті кожного конкретного об'єкта.

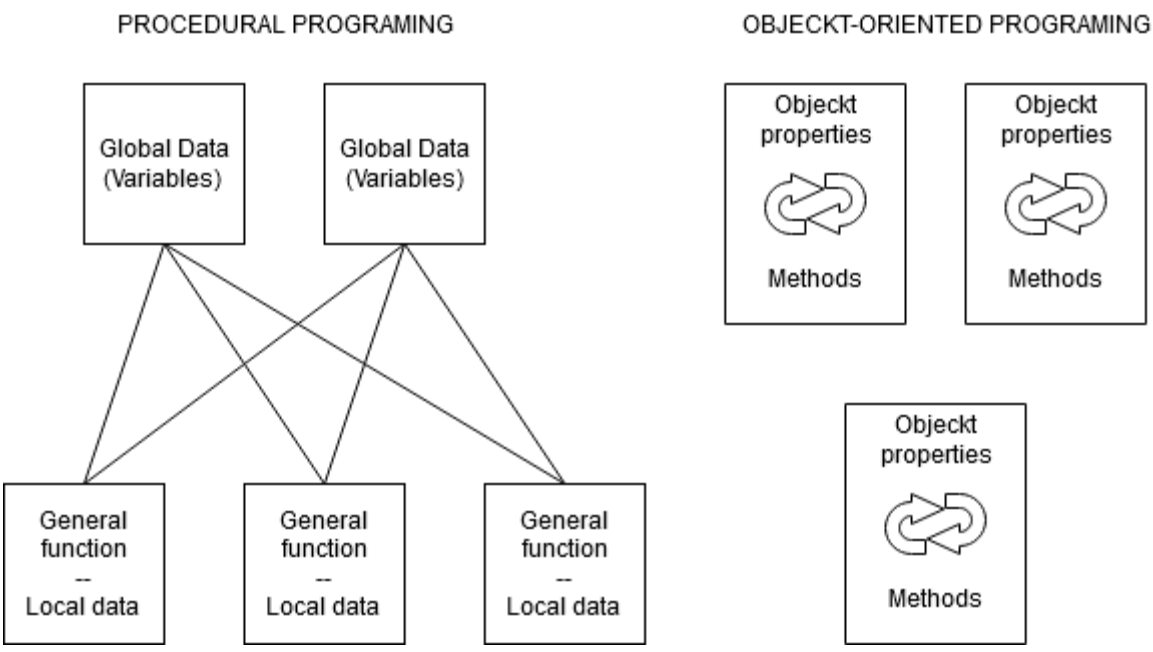


Рис 2.2 Порівняння процедурного та об'єктно-орієнтованого програмування

- Стандарт для корпоративних обчислювальних систем

Корпоративні програми - Java підтримує безліч бібліотек - будівельних блоків будь-якої корпоративної системи. Бібліотеки допомагають розробникам створювати будь-які функції, які можуть знадобитися компанії. Можливості інтеграції Java: більшість хостинг-провайдерів підтримують Java. Більш того, працювати з Java можна з будь-якого комп'ютера, незалежно від конкретної апаратної інфраструктури.

- Незалежність від платформи

Можна створити Java-додаток на Windows, скомпілювати його в байт код і запустити його на будь-який інший платформі, яка підтримує віртуальну машину Java (JVM). Таким чином, JVM служить рівнем абстракції між кодом і обладнанням. Всі основні операційні системи, включаючи Windows, Mac OS і Linux, підтримують JVM. Якщо ваша програма не спирається на специфічні для платформи функції і призначений для користувача інтерфейс, її можна з легкістю перенести: принаймні, більшу її частину.

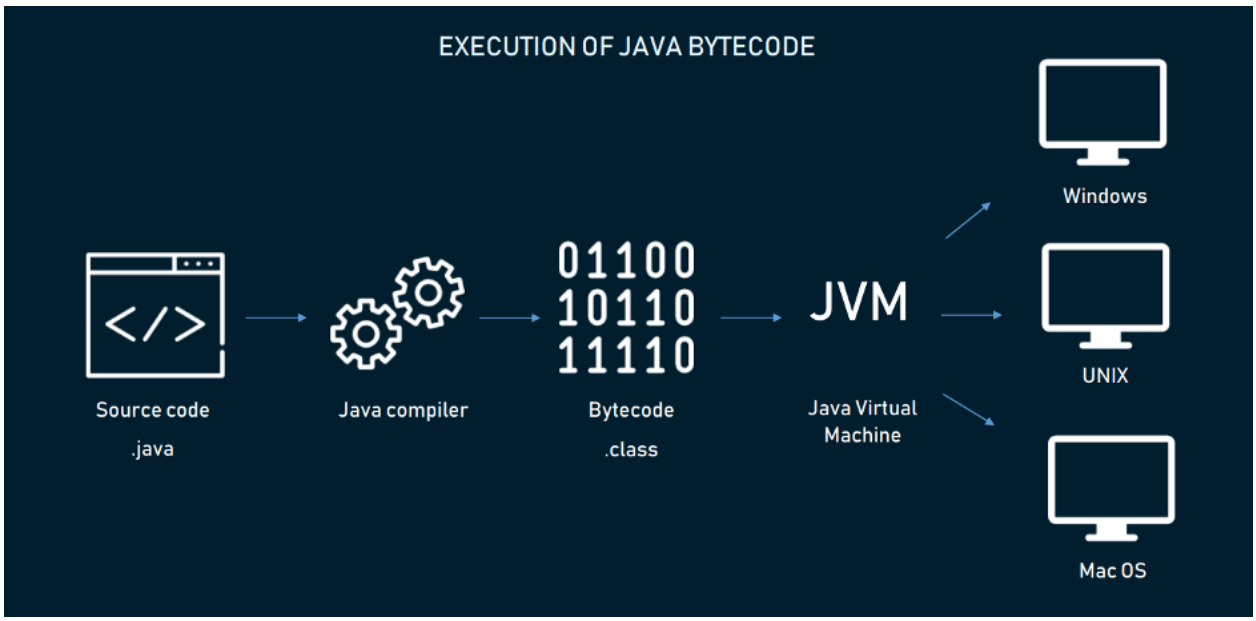


Рис 2.3 Принцип роботи мови Java[22]

- Автоматичне управління пам'яттю

Java не потрібно вручну писати код для управління пам'яттю завдяки автоматичному управлінню пам'яттю (AMM). Ефективність програми безпосередньо пов'язана з пам'яттю. При цьому обсяг пам'яті обмежений. При написанні програми на мовах з ручним керуванням пам'яттю, розробники ризикують забути виділити пам'ять, що призведе до збільшення обсягу займаної додатком пам'яті і проблем з продуктивністю.

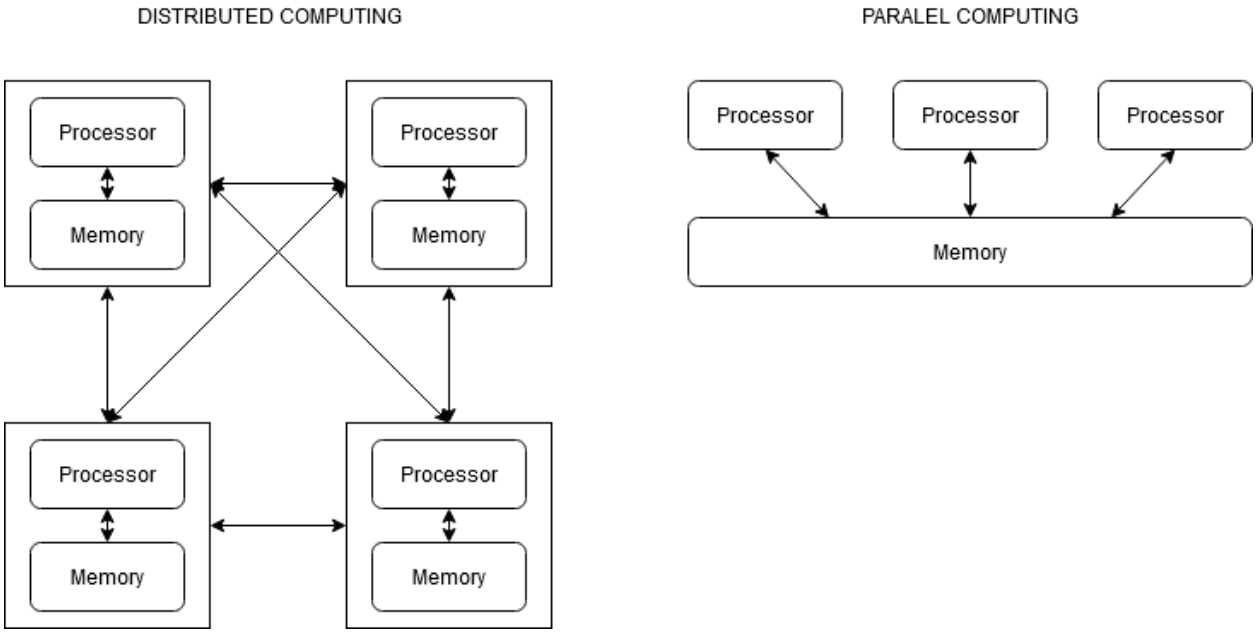


Рис 2.4 Порівняння розподіленого і паралельного програмування

- Багатопотоковість

  Потік — Щоб максимально ефективно використовувати час процесора, Java дозволяє запускати потоки одночасно. Потоки використовують одну і ту ж область пам'яті, тому між ними можна швидко перемикатися. Це особливо корисно програмах з великим об'ємом анімації.

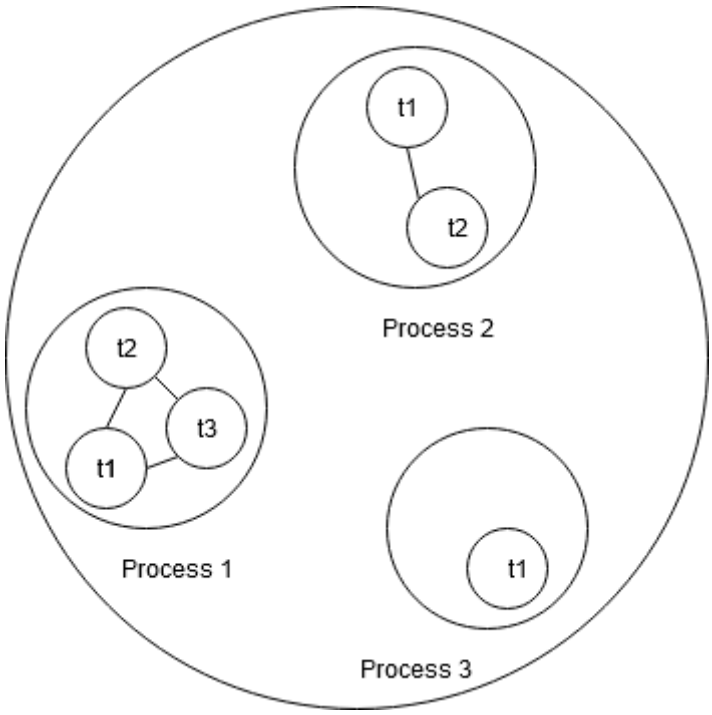


Рис 2.5 Приклад багатопотокового виконання

2.1.3. Вибір допоміжних бібліотек

2.1.3.1 CMake

CMake може обробляти збірки на місці і поза ним, що дозволяє кілька збірок з одного дерева джерел і перехресну компіляцію . Можливість побудови дерева каталогів поза деревом-джерелом є ключовою особливістю, гарантуючи, що якщо каталог збірки буде видалено, вихідні файли залишаються незадіяними. може знаходити виконувані файли, файли та бібліотеки. Також добре підтримує складні ієрархії каталогів та додатків, які покладаються на кілька бібліотек.[21]

2.1.3.2 JOSM

JOSM - це багатофункціональний розширений офлайн-настільний редактор для даних OpenStreetMap.[24]

- Поширені інструменти GIS для робочого столу:
- Настроювання панелі інструментів, контроль перегляду (масштабування, панорамування тощо), управління стилями, значками та шарами.

Відкрити локальні дані:

- Файли NMEA-0183: .nmea, .nme, .nma, .log, .txt
- Файли OSM: .osm, .xlm, .osmbz2, .osmbz
- Файл зміни OSM: .osc, .osc.bz2, .osc.bz, .osc.gz
- зображення (.jpg)

Зображення:

- Візуалізуйте базові карти з OSM, Bingsat, Lansat, супутника MapBox, MapQuest Open Aerial або будь-яких інших джерел WMS.

Інструменти редагування:

- Вузли: об'єднати, об'єднати, видалити клей, розподілити, вирівняти по колу, вирівняти в рядку, приєднати вузол до шляху тощо.
- Способи: розділити, об’єднати, повернути назад, спростити, розклеїти шляхи тощо.
- Області: приєднуйтесь до областей, що перекриваються, створюйте багатокутники тощо.
- Звукове відображення: управління записом обстеження.
- Фотографування: управління фотографіями опитування.
- Плагіни: для завантаження доступний список спеціалізованих плагінів.

2.2. Основні рішення з реалізації додатку та його компонентів

Розробку додатку можна поділити на такі пункти:

1. Реалізація серверної частини;
2. Візуалізації панелі приладів;
3. Візуалізації мап;
4. Реалізація можливості мануального компілювання;

2.2.1. Реалізація серверної частини

Для зручного ведення розробки та можливості масштабування додатку у майбутньому було вирішено поділити систему на три основні компоненти:
- VWS (Virtual World Server) - даний компонент відповідає за обробку сигналів які надсилають у систему IoT девайси. Тобто веде обрахунок оброблених даних та подій які можуть вплинути на загальну систему Smart city. Дана система відповідальна за візуалізацію даних у іншому компоненті системи - OpenStreetMap. Також він дозволяє можливість спілкування між девайсами які підключені до системи Smart city.

- SCS (Smart City Server) - компонент SCS відповідає за обробку сигналів які надсилають у систему автомобілі. Дозволяє транспортним засобам спілкуватися між собою. Також система відповідальна за візуалізацію даних у іншому компоненті системи - OpenStreetMap.

- OpenStreetMap - відповідальний за обробку даних які поступають на нього з VWS і SCS, після якої вносить зміни на мапі.

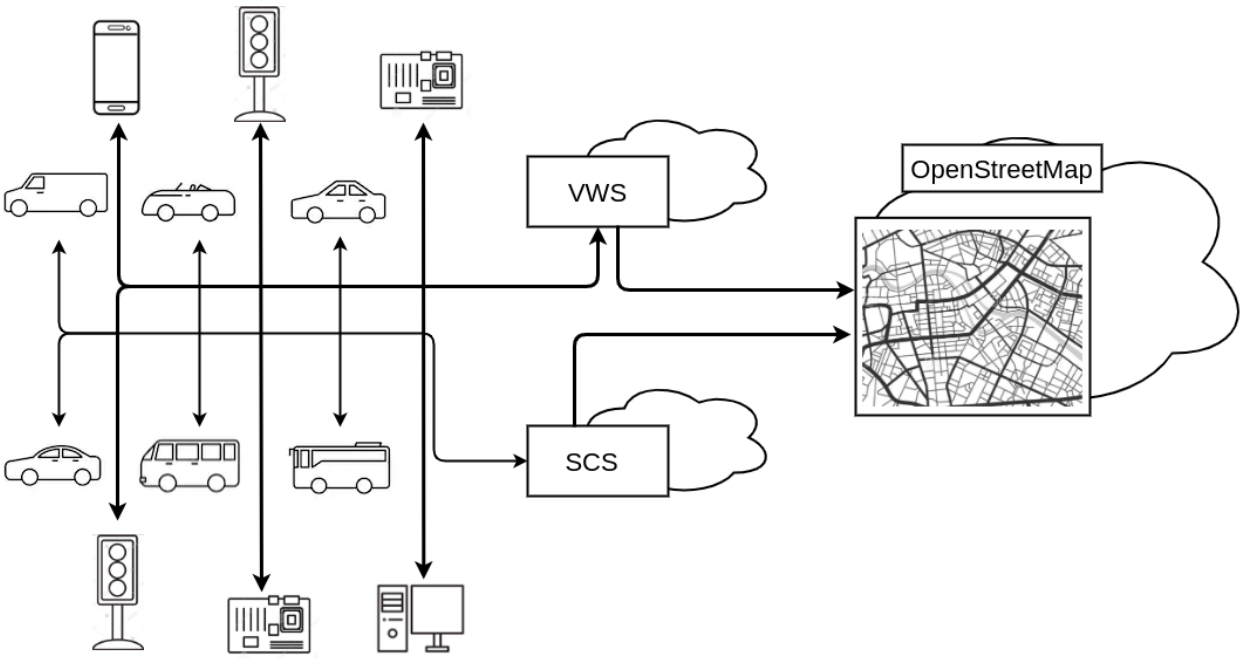


Рис. 2.6. Архітектура роботи системи Smart City

2.2.2. Візуалізації панелі приладів

Для реалізації даного компонента системи було обрано фреймворк з відкритим кодом Qt Automotive Suite, який вже має заготовки для реалізації анімацій панелі приладів. В ході розробки було створено два типи панелей приладів:

- панель гібридного авто див. рис. 2.7.

Дані панелі мають однаковий функціонал в який входять індикатори швидкість руху, кількості оборотів в хвилину двигуна, індикатор заряду акумулятора, індикатор кількості палива та масла, попередження про активоване аварійне гальмо, попередження про не пристібнутий пасок безпеки, попередження про проблеми з двигуном, попередження про пошкодження колес, попередження про низький рівень палива та масла, попередження низького заряду акумулятора, а також сигнал про включені ходові вогні і ввімкнені повороти.

Рис. 2.7. панель приладів гібридного авто

2.2.3. Візуалізації мап

Для реалізації даного компонента системи було обрано систему з відкритим кодом OpenStreetMap, яка має велику кількість фреймворків для редагування та внесення змін в режимі реального часу. Основною задачею даної компоненту є відображення геолокації транспортного засобу та відображення стану доріг, який підключений до системи Smart City.

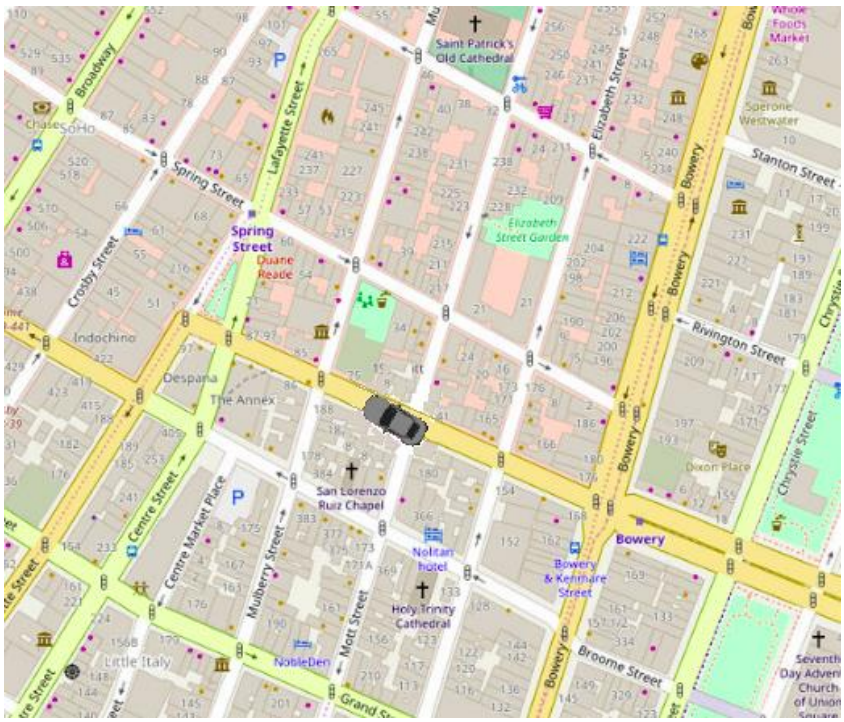


Рис. 2.8. Приклад візуалізації мап

2.3 Бібліотека Boost

Ця бібліотека має набір бібліоткек даля мови C++, яка забезпечує підтримку завдань та структур, таких як лінійна алгебра , генерація псевдовипадкових чисел , багатопоточність, обробка зображень , регулярні вирази та тестування одиниць .[20]

2.3.1 Опис бібліотеки

- Boost.Algorithm

  Надає різні алгоритми, що доповнюють алгоритми зі стандартної бібліотеки;

- Boost.Any

  Надає тип під назвою boost :: any, який може зберігати об'єкти довільних типів;

- Boost.Array

  Дозволяє обробляти масиви C ++, як контейнери зі стандартної бібліотеки;

- Boost.Asio

  Дозволяє розробляти додатки, такі як мережеві програми, які обробляють дані асинхронно;

- Boost.Assign

  Надає допоміжні функції для додавання декількох значень до контейнера без необхідності повторного виклику функцій-членів, таких як push_back ();

- Boost.Atomic

  Визначає клас boost :: atomic для виконання атомних операцій на інтегральних значеннях. Бібліотека використовується в багатопотокових програмах, яким потрібно ділити цілісні значення між потоками;

- Boost.Bimap

  Надає клас під назвою boost :: bimap, який схожий на std :: map. Важлива відмінність полягає в тому, що boost :: bimap дозволяє шукати як ключ, так і значення;

- Boost.Bind

  Boost.Bind - це адаптер для передачі функцій як параметрів шаблону, навіть якщо підпис функції несумісний із очікуваним параметром шаблону;

- Boost.Chrono

  Визначає численні тактові частоти для отримання таких значень, як поточний час або час процесора;

- Boost.CircularBuffer

  Пропонує круглий контейнер з постійним розміром пам'яті;

- Boost.CompressedPair

  Надає структуру даних boost :: compression_pair, яка схожа на std :: pair, але потребує меншої пам’яті, коли параметри шаблону порожні класи;

- Boost.Container

  Визначає всі контейнери зі стандартної бібліотеки, а також додаткові контейнери, такі як boost :: container :: list;

- Boost.Conversion

  Надає двом операторам литового режиму виконувати пониження та перехресні касти;

- Boost.Coroutine

  Дає можливість використовувати підпрограми в C ++. Супроводи часто використовуються в інших мовах програмування, використовуючи ключове слово урожай;

- Boost.DateTime

  Може використовуватися для обробки, читання та запису значень дати та часу;

- Boost.DynamicBitset

  Забезпечує структуру, схожу на std :: bitset, за винятком того, що вона налаштована під час виконання;
- Boost.EnableIf

  Дозволяє перевантажувати функції на основі властивостей типу;
- Boost.Exception

  Дозволяє додавати додаткові дані до викинутих винятків, щоб ви могли надати більше даних для лову обробників;
- Boost.Filesystem

  Забезпечує клас для обробки шляхів і декількох функцій для доступу до файлів і каталогів;
- Boost.Flyweight

  Полегшує використання однойменної моделі дизайну;
- Boost.Foreach

  Надає макрос, подібний до діапазону для циклу, що вводиться з C ++ 11;
- Boost.Format

  Замінює функцію std :: printf () на безпечний і розширюваний клас, формат boost ::;
- Boost.Function

  Спрощує визначення вказівників функції;
- Boost.Fusion

  Дозволяє створювати неоднорідні контейнери - контейнери, в яких можна зберігати елементи різних типів;
- Boost.Graph

  Надає алгоритми для таких операцій, як пошук найкоротшого шляху між двома точками в графіку;
- Boost.Heap

  Надає багато варіантів класу std :: prior_queue зі стандартної бібліотеки;

- Boost.Integer

  Назначає спеціалізовані типи для цілих чисел, які були доступні розробникам C з моменту виходу стандарту C99 у 1999 році;

- Boost.Interprocess

  Використовує спільну пам'ять, щоб допомогти програмам швидко та ефективно спілкуватися;

- Boost.Intrusive

  Визначає контейнери, які забезпечують більш високу продуктивність, ніж контейнери зі стандартної бібліотеки, але які також мають особливі вимоги до об'єктів, які вони містять;

- Boost.IOStreams

  Забезпечує потоки та фільтри. Фільтри можуть бути з'єднані з потоком, наприклад, для запису стислих даних;

- Boost.Lambda

  Дозволяє визначати анонімні функції без C ++ 11;

- Boost.LexicalCast

  Надає оператору передачі для перетворення чисел у рядок і навпаки;

- Boost.Lockfree

  Визначає безпечні для потоків контейнери, до яких одночасно може отримати доступ декілька потоків;

- Boost.Log Boost.Log

  Boost.Log Boost.Log - це бібліотека журналів у Boost;

- Boost.MetaStateMachine

  Дозволяє розробляти державні машини, як вони визначені в UML;

- Boost.MinMax

  Забезпечує алгоритм, який може знайти найменші та найбільші значення в контейнері без виклику std :: min () та std :: max ();

- Boost.MPI

  Забезпечує інтерфейс C ++ для стандарту MPI;

- Boost.MultiArray

  Спрощує роботу з багатовимірними масивами;
- Boost.MultiIndex

  Дозволяє визначати нові контейнери, які можуть підтримувати декілька інтерфейсів, наприклад, такі з std :: vector та std :: map;
- Boost.NumericConversion

  Надає оператору лиття для безпечного перетворення між значеннями різних числових типів, не створюючи умови переповнення;
- Boost.Operators

  Дозволяє автоматично перевантажувати багато операторів за допомогою вже визначених операторів;
- Boost.Options

  Забезпечує клас для позначення необов'язкових повернутих значень. Функції, які не завжди можуть повернути результат, більше не потрібно використовувати спеціальні значення, наприклад -1 або нульовий покажчик;
- Boost.Parameter

  Дозволяє передавати параметри таким функціям, як пари імен / значень, як ви можете, з мовами програмування, такими як Python;
- Boost.Phoenix

  Дозволяє створювати лямбда-функції без C ++ 11. На відміну від функцій C ++ 11 лямбда, функції лямбда з цієї бібліотеки можуть бути загальними;
- Boost.PointerContainer

  Надає контейнери, оптимізовані для управління динамічно виділеними об'єктами;

- Boost.Pool

  Boost.Pool - це бібліотека для управління пам’яттю. Наприклад, Boost.Pool визначає розподільник, оптимізований для ситуацій, коли вам потрібно створити та знищити багато об'єктів, однакового розміру;

- Boost.ProgramOptions

  Дозволяє програмі визначати та оцінювати параметри командного рядка;

- Boost.PropertyTree

  Забезпечує контейнер, який зберігає пари ключів / значень у структурі, подібній до дерева. Це полегшує управління типом даних конфігурації, які використовуються багатьма програмами;

- Boost.Random

  Забезпечує генератори випадкових чисел;

- Boost.Range

  Вводить концепцію під назвою діапазон, яка замінює ітератори, що зазвичай отримуються з контейнерів, з початком () і кінцем (). Діапазон робить це так, що вам не потрібно передавати пару ітераторів алгоритмам;

- Boost.Ref

  Надає адаптери, які дозволяють передавати посилання на об'єкти, які неможливо скопіювати, до функцій, що передають параметри при копіюванні;

- Boost.Regex

  Надає функції пошуку рядків з регулярними виразами;

- Boost.ScopeExit

  Надає макроси для визначення блоків коду, які виконуються, коли закінчується поточна область дії. Таким чином, ресурси можуть бути звільнені в кінці поточного обсягу без використання розумних покажчиків або інших класів;

- Boost.Serialization

  Дозволяє серіалізувати об'єкти та зберігати їх у файлах, які будуть завантажені пізніше;
- Boost.Signals2

  Boost.Signals2 - це рамка для обробки подій на основі концепції сигнал / слот, яка пов'язує функції з сигналами і автоматично викликає відповідну функцію (функції) при спрацьовуванні сигналу;
- Boost.SmartPointers

  Забезпечує набір розумних покажчиків, які спрощують управління динамічно виділеними об'єктами;
- Boost.Spirit

  Дозволяє генерувати парсери, використовуючи синтаксис, аналогічний EBNF (Extended Backus-Naur-Form) ;
- Boost.StringAlgorithms

  Забезпечує безліч автономних функцій для полегшення обробки рядків;
- Boost.Swap

  Визначає boost :: swap (), який має ту ж функцію, що і std :: swap (), але оптимізований для багатьох бібліотек Boost;
- Boost.System

  Пропонує основу для обробки коду помилок, характерних для системи та додатків;
- Boost.Thread

  Дозволяє розробляти багатопотокові програми;
- Boost.Timer

  Визначає годинник, який дозволяє вимірювати продуктивність коду;
- Boost.Tokenizer

  Дозволяє переглядати маркери в рядку;
- Boost.Tribool

  Забезпечує тип, який, на відміну від bool, розрізняє три, а не два стани;

- Boost.Tuple

  Надає узагальнену версію std :: пар, яка може зберігати довільне число значень, а не лише два;
- Boost.TypeTraits

  Забезпечує функції перевірки властивостей типів;
- Boost.Unordered

  Надає два хеш-контейнери: boost :: unordered_set та boost :: unordered_map;
- Boost.Utility

  Boost.Utility - це сукупність різноманітних інструментів, які занадто малі, щоб мати власні бібліотеки і не вміщуються в іншій бібліотеці;
- Boost.Uuid

  Визначає клас boost :: uuids :: uuid та генератори для створення UUID;
- Boost.Variant

  Дозволяє визначати типи, які, як об'єднання, групують кілька типів;
- Boost.Xpressive

  Дозволяє шукати рядки з регулярними виразами. Регулярні вирази кодуються як код C ++, а не як рядки;

ВИСНОВОК ДО РОЗДІЛУ 2

У розділі 2 були, складені основні функції та вимоги до функціоналу додатку. Було обрано та проведено аналіз бібліотек які було вирішено використати при розробці програмного та графічного інтерфейсів системи, Cmake, Boost та JOSM відповідно.

Було побудовано діаграму прецедентів для користування основних функцій сервісу:

1. Можливість швидко розгорнути середу розробки;

2. Доступ до документації по розробці;

3. Можливість швидкого запуску;

4. Можливість мануально компілювання системи;

5. Низькі вимогу до аппаратного обладнення;

6. Можливість вести розробку без доступу до інтернета;

7. Організація розробки группами;

Графічний інтерфейс додатку було розроблено за допомогою ReactJS та OpenStreetMap.

РОЗДІЛ 3.
РОЗРОБКА СИСТЕМИ

3.1 Реалізація можливості мануального компілювання

При розробці додатку було використано велику кількість бібліотек визначеної версії, що у майбутньому при відсутності мануалу по компілюванню може призвести до неможливості роботи із системою, через було вирішено створити детальний мануал по зборці проекту.

3.1.1 Підготовка віртуальної машини

Даний пункт зборки проекта є не обов’язковим якщо компілювання виконується не у віртуальному середовищі:

- VirtualBox 6.0, 20GB disk, 4GB mem
- Ubuntu ubuntu-16.04.6-desktop-amd64
- in the VirtualBox Devices menu, choose Insert Guest Additions CD image & run it & select Devices->Drag and drop->Bidirectional & reboot
- update ubuntu software, but do not upgrade
- Connect host to vpn with cisco client

3.1.2 Встановлення залежностей

Даний пункт є обов’язковим як для компілювання у не у віртуальному середовищі і віртуальному середовищі.

- Відкрити командну строку і ввести команду зображену на рисунку 3.1.

```
root@f0bffc8d8cd5:/# sudo apt install autoconf automake autotools-dev build-e
ssential cmake doxygen freeglut3 freeglut3-dev gcc git libagg-dev libcairo2-d
ev libfreetype6-dev libmarisa-dev libpangocairo-1.0-0 libpango1.0-dev libprot
obuf-dev libqt5svg5-dev libtool libxml2-dev make pkg-config protobuf-compiler
 subversion libgoogle-glog-dev qtbase5-dev libqt5gui5 qttools5-dev-tools qtto
ols5-dev qtdeclarative5-dev qtlocation5-dev qtpositioning5-dev qt5-default qm
l-module-qtquick-* qml-module-qtgraphicaleffects qml-module-qtpositioning lib
boost-thread-dev libboost-system-dev
```

Рис 3.1

3.1.3Генерування генерування ssh ключа для завантаження проекту

```
root@f0bffc8d8cd5:/# ssh-keygen -t ecdsa && cat ~/.ssh/id_ecdsa.pub
```

Рис 3.2

3.1.4 Налаштування середовища git

```
root@f0bffc8d8cd5:/# git config --global user.name "username" &&
 git config --global user.email "username@email.com"
```

Рис 3.3

3.1.5 Створення дерикторії для завантаження проекту

```
root@f0bffc8d8cd5:/# mkdir ~/SBC && cd ~/SBC
```

Рис 3.4

3.1.6 Завантаження проекту з веб сервісу

```
root@f0bffc8d8cd5:/# git clone sbcx/sbc-platform.git
```

Рис 3.5

3.1.7. Ініціалізація проекту

```
root@f0bffc8d8cd5:/# cd sbc-platform && git checkout -b master_lo
cal remotes/origin/master && git submodule update --init
```

Рис 3.6

3.1.8 Завантаження libosmcout

```
root@f0bffc8d8cd5:/# cd ~/Downloads && wget master/lastSuccessful
Build/artifact/package/libosmscout-0.1.20.deb
```

Рис 3.7

3.1.9 Встановлення libosmscout

```
root@f0bffc8d8cd5:/# sudo apt install ./libosmscout-0.1.20.deb
```

Рис 3.8

3.1.10 Налаштування змінних оточення модуля mapviewer

```
root@f0bffc8d8cd5:/# nano ~/SBC/sbc-platform/src/SCS/conf/config.
json && insert real path, for ex. (/home/username/SBC/sbc-platfor
m/src/) instead <FULL_PATH_TO_REPOSITORY>
```

Рис 3.9

3.1.11 Компілювання модуля virtual world server

```
root@f0bffc8d8cd5:/# cd ~/SBC/sbc-platform/src/VWS && mkdir build
 && cd build && cmake .. && make
```

Рис 3.10

3.1.12 Компілювання та встановлення IC-LIB

```
root@f0bffc8d8cd5:/# cd ~/SBC/sbc-platform/src/IC-Lib/ && mkdir b
uild && cd build && cmake .. && make
```

Рис 3.11

3.1.13 Компілювання та встановленя IC_LINUX

```
root@f0bffc8d8cd5:/# sudo make install && sudo ldconfig -v IC_LIN
UX && cd ~/SBC/sbc-platform/src/IC-Linux/ && mkdir build && cd bu
ild && qmake .. && make
```

Рис 3.12

3.1.14 Запуск скомпільованого модуля Virtual world server

```
root@f0bffc8d8cd5:/# cd ~/SBC/sbc-platform/src/VWS/build/ && ./vw
-server -p 40881 -s ../conf/demo-gps.json -t 10 -v
```

Рис 3.13

3.1.15 Запуск скомпільованого модуля mapviewer

```
root@f0bffc8d8cd5:/# cd ~/SBC/sbc-platform/src/SCS/ && ./mapviewe
r/build/SBCStat conf/config.json && cd ~/SBC/sbc-platform/src/IC-
Linux && ./build/SmartCity_IC conf/config.json
```

Рис 3.14

3.2.Структура додатку

```
function parseMapParams(qmap) {
  if (!qmap) return false;
  const params = qmap.split('/').map(Number);
  if (params.length < 3 || params.some(isNaN)) return false;
  return geoExtent([params[2], params[1]]);  // lon,lat
}

const hash = utilStringQs(window.location.hash);
const requested = hash.background || hash.layer;
let extent = parseMapParams(hash.map);

return _loadPromise = ensureImageryIndex()
  .then(imageryIndex => {
    const first = imageryIndex.backgrounds.length && imageryIndex.backgrounds[0];

    let best;
    if (!requested && extent) {
      best = background.sources(extent).find(s => s.best());
    }

    // Decide which background layer to display
    if (requested && requested.indexOf('custom:') === 0) {
      const template = requested.replace(/^custom:/, '');
      const custom = background.findSource('custom');
      background.baseLayerSource(custom.template(template));
      prefs('background-custom-template', template);
    } else {
      background.baseLayerSource(
        background.findSource(requested) ||
        best ||
        background.findSource(prefs('background-last-used')) ||
        background.findSource('Bing') ||
        first ||
        background.findSource('none')
      );
    }
```

Рис.3.15

На фрагменті коду, що вказаний вище function parseMapParam виконує розбиття мапи на шматки в залежності від степені приближення до неї та шикується в спільний паттерн, після нумерація шматкіф хешується.

Після в залежності від при от привязки точки леєру мапи прив'язується шматок бекграунду з раніше визначеного патерну паттерна. Ці всі дії потрібні для корректного відображення інформації, та зручного користування для споживача.

```
function background(selection) {
  const currSource = baseLayer.source();

  // If we are displaying an Esri basemap at high zoom,
  // check its tilemap to see how high the zoom can go
  if (context.map().zoom() > 18) {
    if (currSource && /^EsriWorldImagery/.test(currSource.id)) {
      const center = context.map().center();
      currSource.fetchTilemap(center);
    }
  }

  // Is the imagery valid here? - #4827
  const sources = background.sources(context.map().extent());
  const wasValid = _isValid;
  _isValid = !!sources.filter(d => d === currSource).length;

  if (wasValid !== _isValid) {      // change in valid status
    background.updateImagery();
  }
```

Рис 3.16

На рисунку 3.16 фрагмент коду, на якому відбувається зміна бекграунду мапи в залежності від степені наближення з подальшою перевіркою степені наближення в залежності від якої визначається зміна бекграунду.

```
export function actionDeleteWay(wayID) {

    function canDeleteNode(node, graph) {
        // don't delete nodes still attached to ways or relations
        if (graph.parentWays(node).length ||
            graph.parentRelations(node).length) return false;

        var geometries = osmNodeGeometriesForTags(node.tags);
        // don't delete if this node can be a standalone point
        if (geometries.point) return false;
        // delete if this node only be a vertex
        if (geometries.vertex) return true;

        // iD doesn't know if this should be a point or vertex,
        // so only delete if there are no interesting tags
        return !node.hasInterestingTags();
    }
```

Рис. 3.17

На рисунку 3.16 фрагмент коду, на якому відбувається видалення прокладеного маршруту з можливістю видалення початкової або кінцевої точки.

На рисунку 3.18 фрагмент коду, на якому відбувається додавання та створення кастомних позначень на мапі з можливістю їх редагування.

```
function drawEditable(difference, extent) {
    var mode = context.mode();
    var graph = context.graph();
    var features = context.features();
    var all = context.history().intersects(map.extent());
    var fullRedraw = false;
    var data;
    var set;
    var filter;
    var applyFeatureLayerFilters = true;

    if (map.isInWideSelection()) {
        data = [];
        utilEntityAndDeepMemberIDs(mode.selectedIDs(), context.graph()).forEach(function(id) {
            var entity = context.hasEntity(id);
            if (entity) data.push(entity);
        });
        fullRedraw = true;
        filter = utilFunctor(true);
        // selected features should always be visible, so we can skip filtering
        applyFeatureLayerFilters = false;

    } else if (difference) {
        var complete = difference.complete(map.extent());
        data = Object.values(complete).filter(Boolean);
        set = new Set(Object.keys(complete));
        filter = function(d) { return set.has(d.id); };
        features.clear(data);

    } else {
        // force a full redraw if gatherStats detects that a feature
        // should be auto-hidden (e.g. points or buildings)..
        if (features.gatherStats(all, graph, _dimensions)) {
            extent = undefined;
        }

        if (extent) {
            data = context.history().intersects(map.extent().intersection(extent));
            set = new Set(data.map(function(entity) { return entity.id; }));
            filter = function(d) { return set.has(d.id); };
```

Рис. 3.18

ВИСНОВКИ

Розробка даного дипломного проекту була присвячена дослідженню можливих варіантів рішень, спрямованих на вирішення задачі створення автопілоту адаптованого під систему Smart City.

В ході виконання проекту були розглянуті вже існуючі на даний момент рішення, які реалізують поставлені задачі. На основі цих рішень було складено порівняльну характеристику з урахуванням всіх переваг та недоліків кожного. Було запропоновано варіанти модернізації даної системи, які включають в себе комплексне вирішення основних проблем існуючих додатків.

Також було проаналізовано предметну область даної роботи та визначено основні вимоги і функції додатку. На основі визначених вимог до системи побудовані схематичні рисунки та таблиці. Проведено проектування інтерфейсу та побудовано відповідні компоненти візуалізації системи.

Зважаючи на вище задані вимоги був проведений аналіз використання існуючих технологій та платформ для реалізації системи. Оптимальною платформою для реалізації було обрано Unix подібну систему Ubuntu 20.04 із сімейства Linux спираючись на економічну та функціональну вигідність використання. За допомогою технологій Docker і Docker-Compose додаток був реалізований з можливістю розгортання на більшості платформ Linux, Windows, Mac Os - відповідно. Також було обрано мову написання проекту, і проведений опис використаних додаткових бібліотек з обґрунтуванням доцільності їх використання.

Реалізація додатку відбувалася у 5 основних етапи з урахуванням зазначеного у технічному завданні функціоналу та вимог до розробки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. GPS посилланя на веб ресурс → www.itrack.com.ua/ua/support/docs/section-3 GPS
2. V2X посилланя на веб ресурс → https://en.wikipedia.org/wiki/Vehicle-to-everything V2X
3. V2C посилланя на веб ресурс → https://contest.techbriefs.com/2018/entries/automotive-transportation/9094-0701-161849-vehicle-to-cloud-v2c-remote-management-of-electric-hybrid-and-plug-in-hybrid-electric-vehicle-charging
4. З'язок взаємодії V2X посилланя на веб ресурс → http://168.131.150.78/xe/index.php?mid=V2X&document_srl=514
5. V2C посилланя на веб ресурс → https://en.wikipedia.org/wiki/Connected_car
6. 3GPP посилланя на веб ресурс → https://en.wikipedia.org/wiki/3GPP
7. DSRC посилланя на веб ресурс → https://en.wikipedia.org/wiki/Dedicated_short-range_communications
8. GPS посилланя на веб ресурс → https://en.wikipedia.org/wiki/Global_Positioning_System
9. OCX посилланя на веб ресурс → https://www.raytheonintelligenceandspace.com/capabilities/products/gps-ocx
10. NAT посилланя на веб ресурс → https://www.juniper.net/documentation/en_US/junos/topics/topic-map/security-nat-overview.html
11. NAT Cloud посилланя на веб ресурс → https://cloud.google.com/nat/docs/overview
12. OpenStreetMap посилланя на веб ресурс → https://www.hellojae.com/home/open-street-map

13. OpenStreetMap посилланя на веб ресурс →
https://en.wikipedia.org/wiki/OpenStreetMap
14. JOSM посилланя на веб ресурс → https://en.wikipedia.org/wiki/JOSM
15. ID Software посилланя на веб ресурс →
https://en.wikipedia.org/wiki/ID_(software) ID software
16. Qt Automotive Suite посилланя на веб ресурс →
https://doc.qt.io/QtAutomotiveSuite/index.html#components-and-tools
17. Boost Library посилланя на веб ресурс →
https://en.wikipedia.org/wiki/Boost_(C++_libraries)
18. Візуалізація мапи OpneStre посилланя на веб ресурс →
https://www.hellojae.com/home/open-street-map
19. Java пирлади роботи та порівняння посилланя на веб ресурс →
https://medium.com/nuances-of-programming/плюсы-и-минусы-программирования-на-java-2861f4c2a0d5
20. Cmake посилланя на веб ресурс → https://en.wikipedia.org/wiki/CMake
21. JOSM посилланя на веб ресурс →
https://live.osgeo.org/en/overview/josm_overview.html

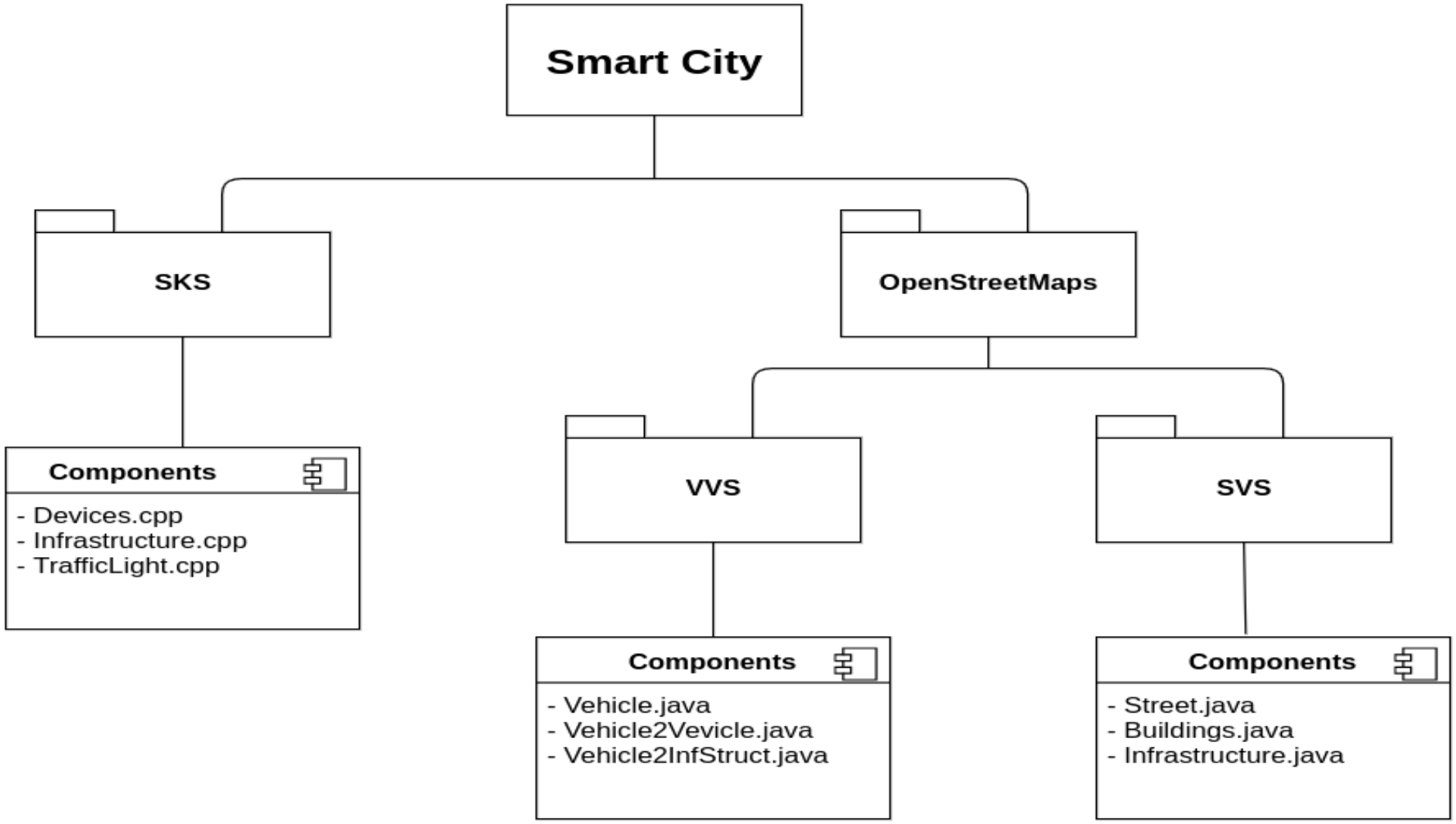
ДОДАТОК 1

Система відображення стану доріг.

Схема структурна – структура програми
ІАЛЦ.467800.004 Д1

Аркушів 1

Київ – 2020 р.



					ІАЛЦ.467800.004 Д1
Зм.	Арк.	№ докум.	Підпис	Дата	
Розробив		Диренков В.О.			Система відображення стану доріг. Схема структурна
Перевірив		Стешин В.В.			
Реценз.					
Н. Контр.		Сімоненко В.П.			
Затв.		Стіренко С.Г.			

Літ.	Аркуш	Аркушів
	1	1

НТУУ «КПІ», ФІОТ, ІО-63

ДОДАТОК 2

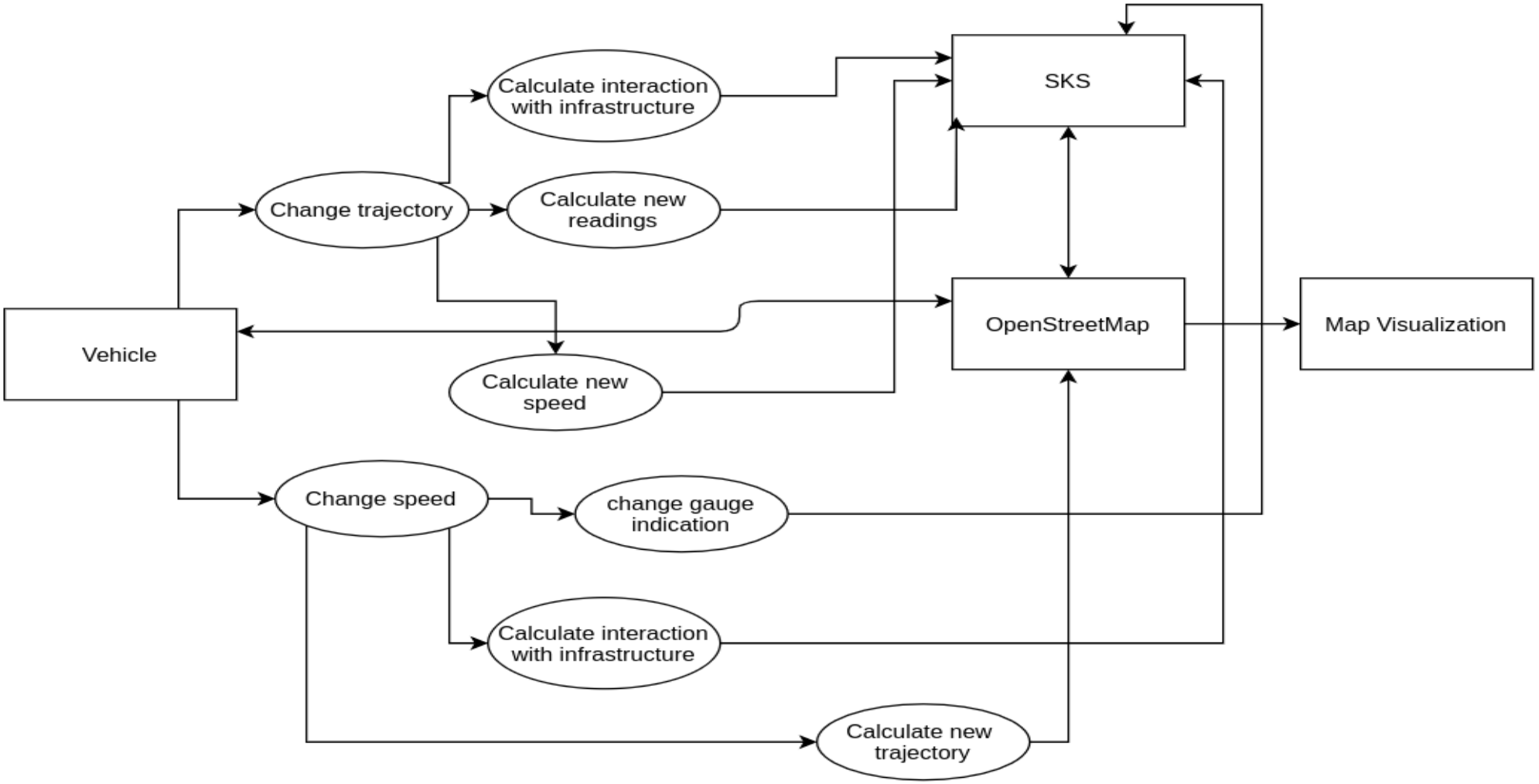
Система відображення стану доріг

Схема функціональна – схема прецедентів

ІАЛЦ.467800.005 Д2

Аркушів 1

Київ – 2020



					ІАЛЦ.467800.005 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Диренков В.О.			Система відображення стану доріг. Схема прецедентів	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Стешин В.В.				
Реценз.						НТУУ «КПІ», ФІОТ, ІО-63
Н. Контр.		Сімоненко В.П.				
Затв.		Стіренко С.Г.				

ДОДАТОК З

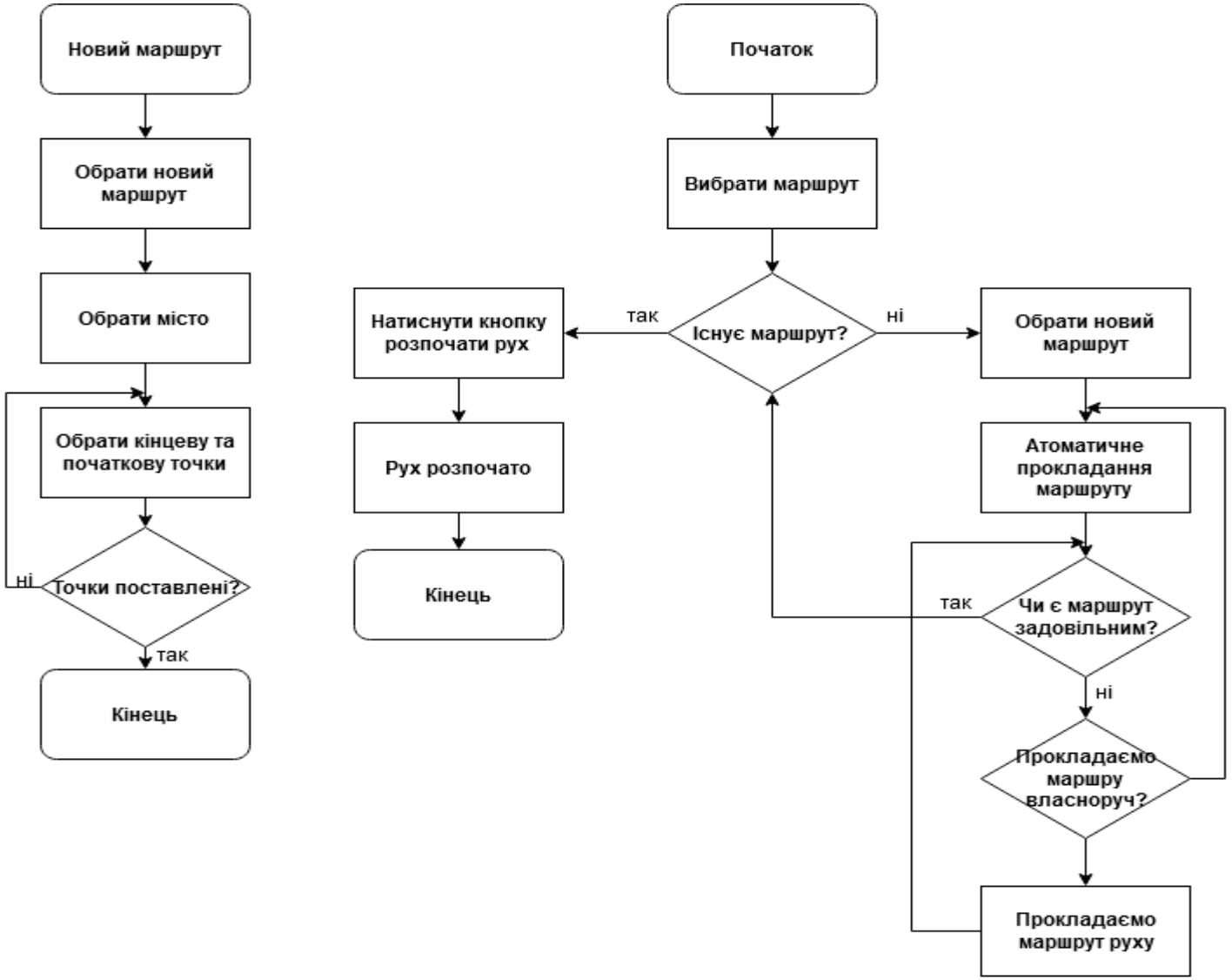
Система відображення стану доріг

Схема принципова – схема алгоритму додавання нового пристрою

ІАЛЦ.467800.006 Д3

Аркушів 1

Київ – 2020



					ІАЛЦ.467800.006 ДЗ	
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розробив		*Диренков В.О.*			*Система відображення стану доріг. Схема алгоритму додавання нового пристрою*	*Літ.* / *Аркуш* 1 / *Аркушів* 1
Перевірив		*Стешин В.В.*				
Реценз.						*НТУУ «КПІ», ФІОТ, ІО-63*
Н. Контр.		*Сімоненко В.П.*				
Затв.		*Стіренко С.Г.*				